Автоматное программирование технологических контроллеров на языке Forth

А.П. Бурков, Е.В. Красильникъянц (Ивановский государственный энергетический университет им. В.И. Ленина)

Рассмотрены вопросы, связанные с применением способов программирования технологических контроллеров промышленных установок, основанных на концепции автоматного программирования. Предложена реализация данной концепции на языке Forth, обеспечивающая возможность создания простого, компактного, самодокументируемого кода программ, удобного для реализации и отладки. Приведен пример применения данного подхода при программировании контроллера следящего привода.

Ключевые слова: автоматное программирование, состояние автомата, компиляция, контроллер, управление, самодокументируемый код.

Введение

В настоящее время существует множество языков программирования контроллеров технологических установок. Стандарт IEC61131–3 предусматривает пять основных типов языков программирования, применяемых для поточных линий, технологических агрегатов, роботов и подобных устройств. К ним относятся языки: лестничных диаграмм LD, диаграмм функциональных блоков FB, последовательных функциональных схем SFC, типа «структурированный текст» ST, списка инструкций IL. Характерными особенностями таких языков являются:

– возможность простой модификации программ и наращивания функциональности;

– переносимость проектов с одного ПЛК на другой;

– возможность повторного использования отработанных фрагментов программ.

Считается, что эффективность подготовки программ повышается при использовании специальных сред разработки (IDE), которые содержат все необходимые средства для написания, тестирования и отладки программ с помощью эмуляторов и реальных ПЛК, а также множество готовых фрагментов программного кода.

Однако для низкоуровневых систем, работающих, как правило, в режиме автоматического управления и имеющих большое число нестандартных устройств ввода/вывода, применение стандартных широко распространенных средств разработки имеет ряд недостатков. Ввиду того, что производители стремятся охватить как можно больший круг пользователей, на практике используется лишь небольшая часть IDE, а большая часть дорогостоящей системы не используется. При этом ряд требуемых пользователю функций в среде разработки отсутствует, и их приходится моделировать с помощью IDE, что приводит к усложнению программ и в ряде случаев весьма затруднительно. Обычно решение этой проблемы разработчики IDE видят в создании новых систем и языков программирования.

Вместе с тем существует и альтернативный вариант — расширение функционального состава уже существующих средств разработки. Для реализации этой концепции используемое ПО должно иметь собственные средства компиляции, что позволит непрерывно расширять функциональный состав языков и средств разработки. Среди языков, оснащенных средствами «отложенной компиляции», отметим Forth.

Применение языка Forth в технологических контроллерах

Анализ технической литературы показывает, что в качестве основы собственных систем разработки нередко используется язык Forth, вполне соответствующий перечисленным выше особенностям и достаточно широко распространенный в США и Западной Европе. Язык основан на концепции виртуальной Forth-машины — простого механизма передачи управления потоку команд [1]. Достоинствами языка является простота и компактность, возможность работы в интерактивном режиме и наличие свободно распространяемого ПО.

Язык программирования Forth появился в начале 1970-х гг. в США. Изобретатель языка Чарльз Мур использовал его для разработки ПО микро-ЭВМ, управляющей работой радиотелескопа. Впоследствии появились: множества версий языка практически для всех современных микропроцессоров [2–4], реализации радиационно-стойких Forth-процессоров в кремнии и даже микросхемы, содержащие до 144 Forth-ядер в одном кристалле http://www.greenarraychips.com.

Большим достоинством языка Forth является возможность отладки программ в интерактивном режиме. Во многих случаях разработка ПО носит итерационный характер, что может быть обусловлено неполнотой описания технологических процессов,

Http://www.avtprom.ru

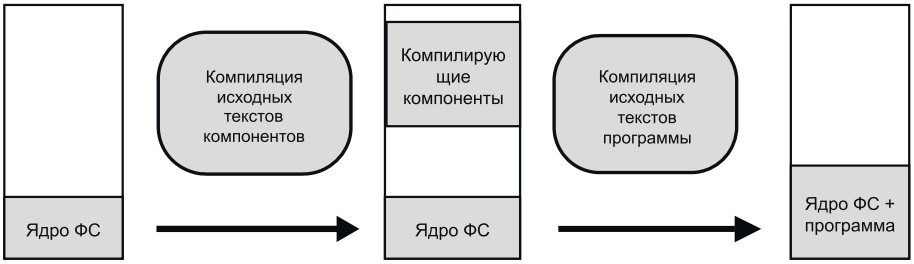

Рис.1. Последовательность компиляции расширений ФС

недостатком технической информации о характеристиках механизмов и несовершенством математических моделей. В этих условиях итеративный процесс разработки сводится к периодическому исследованию характеристик объекта, коррекции ПО, компиляции и отладки даже при наличии формальной спецификации [5]. Именно при этих условиях достоинства Forth сказываются в наибольшей мере.

Вместе с тем Forth имеет и существенный недостаток, заключающийся в том, что стандартный набор операций весьма невелик (около 200 ед.), вследствие чего пользователь должен использовать набор проблемно-ориентированных расширений для решения конкретной задачи. Следует отметить, что языки типов С, С ++ и Pascal/Delphi групп не имеют интерактивного режима и расширяемости, так как никакие изменения компилятора невозможны.

К настоящему времени ряд зарубежных фирм разрабатывает Forth-системы (ФС) для большинства современных микропроцессоров и микроконтроллеров, пригодных для построения технологических контроллеров http://www.mpeforth.com.

Примером являются контроллеры и модули расширения серии ForthLogic™, использующие одноименный язык, разработанные предприятием Електросвіт. Но и в этом случае предлагается использовать стандартный набор средств Forth без каких-либо расширений.

Свойство расширяемости языка может использоваться различным образом. Наиболее простым является написание текстов, непосредственно компилируемых в исполняемый код лишь с помощью основного ядра ФС, расположенного в постоянной памяти контроллера. Главным недостатком этого способа является громоздкость и сложность исходного кода программы, что обусловлено ограниченным набором стандартных операций ядра системы. Для более эффективной работы программиста необходимо существенное расширение набора команд, добавление определений управляющих конструкций и структур данных, основанных на стандартном наборе операций. Такие расширения ядра, обеспечивающие компиляцию, также могут быть выполнены в виде соответствующих текстов. В этом случае процесс компиляции исполняемого кода происходит в два этапа: сначала с помощью основного ядра компилируются программы требуемых расширений, что повышает возможности основного ядра. Затем в объектный код компилируются тексты основной программы (мета компиляция) (рис. 1). После завершения компиляции коды текстов компиляции расширений удаляются из ФС. Изменениям подвергаются только тексты расширений, а состав основного ядра ФС остается неизменным при смене виртуальной Forth-машины и аппаратной платформы. При наличии достаточного объема оперативной памяти компиляция и отладка программ может производиться непосредственно на технологическом контроллере. Использование подобного способа компиляции позволяет обеспечивать совместимость текстов расширений и соответственно программ, выполненных в разное время и на разных аппаратных платформах.

Автоматное программирование на языке Forth

Одним из перспективных методов программирования является использование парадигмы так называемого автоматного программирования [6]. На сегодняшний день разработаны программные средства для стандартных языков программирования, таких как C/C++, текстовых языков, например, Finite State Machines в JavaScript, Finite State Machine в программном комплексе MATLAB фирмы MathWorks и др.

Так, например, IAREmbedded Workbench включает набор удобных в работе инструментов разработки для встроенных приложений: IAR C/C ++ компилятор, ассемблер, компоновщик, библиотекарь, редактор текста, менеджер проектов и отладчик в среде разработки (IDE). В набор может входить графическая среда разработки IARvisualSTATE [7]. Аналогичные средства входят в состав Microsoft Robotics Studio, в язык программирования UML и в ряд других систем.

Широкое распространение указанных средств объясняется тем, что в некоторых случаях затруднительно использование структурированного кода, но достаточно просто проблема решается использованием методов программирования конечных автоматов (FSM). К таким случаям относится и программирование для логических контроллеров, в которых существует необходимость выдачи множества выходных сигналов в зависимости от множества входных, возникающих в произвольном порядке.

Несмотря на то, что программирование и в этих случаях может быть осуществлено через логические выражения (IF, ELSE, THEN), получаемый код трудно читаем, сложен для изменения и отладки.

Язык Forth с помощью средств мета компиляции позволяет получать естественные, удобочитаемые выражения для FSM. В связи с этим в технической литературе представлены варианты построения таких автоматов на Forth [8].

Описание автомата на языке Forth представляет собой набор Forth-слов, состоящий из управляющих конструкций и определений. Автомат может находиться только в одном состоянии из всех возможных. Переход из одного состояния в другое может происходить лишь в случае, если одно из условий перехода является истинным. При переходе в новое состояние могут быть выполнены некоторые действия.

Имена состояний, условных переходов и действий могут назначаться произвольно, но, как правило, они должны отражать семантику исполнения, что обеспечивает свойство самодокументируемости программы. Имена сохраняются и в скомпилированном коде, что вместе с содержимым таблицы состояний обеспечивает простоту отладки автомата и возможность мониторинга в процессе работы. Все Forth-слова, входящие в состав автомата, должны обладать свойством реентерабельности, то есть исполнение их не должно зависеть от числа повторных вхождений в автомат. В качестве слов или имен состояний могут применяться другие автоматы, в том числе и вложенные.

Общая структура автомата может быть представлена в виде разреженной матрицы, где диагональю матрицы являются состояния размерности N, а внедиагональные элементы — условия переходов из одного состояния в другие (возможно, включающие «действия»).

Число условий переходов не может быть больше чем N (N-1), где N-число состояний автомата. Однако на практике, как правило, число переходов значительно меньше N и матрица автомата является разреженной.

Число состояний автомата и размер матрицы может быть любым, однако следует учитывать, что чрезмерное увеличение числа состояний сильно усложняет анализ и отладку. Опыт показывает, что автомат целесообразно составлять для каждого функционального блока управления с числом состояний ⩽10...15. В случае необходимости использования большего числа состояний автомат может быть разбит на два или большее число автоматов меньшей размерности.

Автомат является Forth-словом, всегда имеет одно начальное состояние и может иметь одно или несколько конечных (терминальных) состояний, из которых нет ни одного перехода в другие состояния.

При вызове Forth-слова с именем автомата всегда выполняется одно из его состояний в следующем порядке:

1) выполняется слово с именем состояния, на которое указывает цифровое значение в таблице состояний;

2) проверяются условия переходов: если хотя бы одно из них верно, то выполняется действие, предусмотренное в данном переходе (при его наличии), и изменяется номер текущего состояния в таблице состояний на номер того состояния, на которое происходит переход;

3) производится выход из автомата.

Рис. 2. Контроллер электропривода 5кВт

Применение автоматов в контроллере следящего привода

В качестве примера использования автоматов рассмотрим описания исходных текстов макетного образца контроллера электропривода мощностью от 5 кВт (рис. 2), разработанного в лаборатории кафедры микроэлектроники Ивановского энергетического университета. Электропривод предназначен для векторного управления вентильными и асинхронными двигателями переменного тока, снабженными энкодерами обратной связи, в системах числового программного управления.

Аппаратной платформой контроллера является микросхема LPC2292. Оперативная память микросхемы, необходимая для компиляции программы, расширена с помощью двух микросхем K6R4016V. Обработка данных от энкодеров, формирование широтно-модулированных сигналов (ШИМ), сигналов датчиков токов и ввода/вывода производится с помощью микросхемы FPGA A3P400PQ208 фирмы Actel. Силовая часть с IGBT-модулями расположена в отдельном блоке, аналогичном используемым в системах с контроллерами PMAC фирмы Delta Tau.

Программное обеспечение контроллера представлено набором текстовых файлов (текстов) и ядром системы в двоичном коде, загруженном в постоянную память LPC2292. Отладка ПО производится с помощью терминальной программы exper.exe, разработанной с помощью IDE Win32Forth https://sourceforge.net/projects/win32forth.

Тексты, загружаемые для компиляции в память контроллера, делятся на системные, предназначенные для расширения возможностей компиляции и конкретные приложения для электропривода. Первые обеспечивают возможность использования создания локальных переменных, структур данных, ассемблерных примитивов, в частности, компиляцию описаний автоматов.

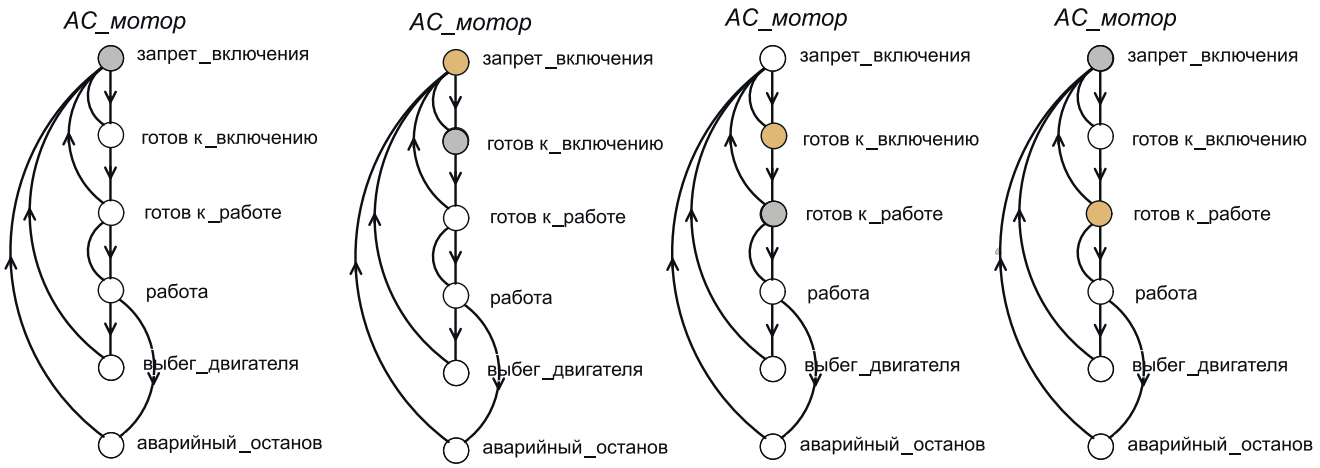

Рис. 3. Изменения диаграммы состояний автомата при выполнении условий переходов «готов_к_включению?», «готов_к_работе?» и «начать_отключение?»

Ядро системы, используемой в контроллере, получено переработкой Forth-системы для платформы Strong Arm на платформу ARM7TDMI и имеет объем 41 Кб. http://www.forth.org.com.

В рассматриваемой работе создан файл приложения **diagr.txt**, содержащий описания трех автоматов — **«тест», «АС_мотор», «сервомотор»**. Остановимся подробнее на описании автомата управления асинхронным двигателем **«АС_мотор»**, имеющего шесть состояний. Графические изображения этого автомата в процессе отладки, сформированные программой **exper.exe**, представлены на рис. 3 в виде расширенных метафайлов ОС Windows. Активное состояние автомата отображается коричневым цветом, прошлые состояние заштрихованы, а неактивные состояния — окружностями белого цвета.

Задавая различные значения условий перехода, проверяют корректность структуры состояний и переходов. Для контроля выполнения действий осуществляют установку и сброс необходимых переменных. Если некоторые комбинации условий переходов ведут к осцилляции автомата, на экране наблюдается мерцание состояний и дуг переходов. В этом случае следует проверить совместимость условий переходов, и если такие условия могут быть совместимы одновременно, следует изменить структуру автомата и/или условия переходов.

Сравнительно небольшая доля текста программного кода, описывающего работу автоматов (26%), в данном случае объясняется значительным объемом слов, использованных в описаниях контуров цифрового управления (53%).

Суммарно для всего комплекса состояний, описывающих работу контроллера электропривода, получены следующие показатели

Объем системных файлов, байт.....42139
Файлов приложения, байт, ...163337
в том числе:
автоматы 53094
регуляторы, драйверы, секция инициализации и пр. 110243
Объем двоичного кода, байт:
Forth-системы.....41392
общий объем.....151456

Выводы

Опыт применения методов автоматного управления в системах управления технологическими установками в течение длительного времени [9] показывает, что автоматное программирование на языке Forth является достаточно простым, экономичным и эффективным методом разработки, отладки и ввода в эксплуатацию программируемых контроллеров. Дополнительные затраты времени на создание средств компиляции достаточно малы по сравнению с выигрышем времени при отладке контроллеров в условиях производства.

Общая доля программного кода, описывающего работу автоматов, в исходных текстах ПО контроллеров технологических установок составила 65...85%. Эти значения показывают, что методы автоматного управления являются хорошей альтернативой использованию дорогостоящих IDE.

Использование принципов метакомпиляции при создании программных шаблонов позволяет обеспечить автоматическое создание областей данных и соответствующих структур управления для драйверов каналов связи, подсистем графического контроля, отладки и автоматизированного тестирования состояний автоматов.

Перспективным является применение методов автоматного управления в многоядерных процессорах, что обеспечит повышение надежности и быстродействия.

Отметим, что применение языка Forth не является единственно возможным способом реализации автоматного управления. Однако его компактность,

наличие функции «отложенной компиляции» и присутствие на рынке большого числа свободно распространяемого ПО является основой для широкого применения данных методов в системах промышленной автоматизации и робототехнике.

Список литературы

1. *Pelc Stephen*. Updating the Forth Virtual Machine. EuroForth. 2008.
2. *Noble J.V.* Scientific FORTH: a modern language for scientific programming. Mechum Banks Publishing, Ivy, VA 1992.
3. *Brodie L.* Thinking FORTH. Englewood Cliffs. Prentice-Hall. 1984.
4. *Hand, T.* The Harris RTX 2000 Microcontroller // Journal of Forth Application and Research. 1990, Vol. 6, No. 1, pp. 5-13.
5. *Pelc Stephen.* Programming Forth. MicroProcessor Engineering Limited. 2005.
6. *Everett F. Carter.* Robots and Finite-State Machines // Dr. Dobb's Journal. 1997. February.
7. *Yiu Joseph, Johnson Ian.* The Many Ways of Programming an ARM® Cortex®-M Microcontroller. 2013. https://www.eeweb.com
8. *Noble J.V.* Finite State Machines in Forth. University of Virginia, Charlottesville. 1995.
9. *Бурков А.П., Красильникъянц Е.В.* Принципы построения программного обеспечения систем управления движением. //Автоматизация в промышленности. 2011. №5.

Бурков Александр Павлович *– канд. техн. наук, ст. научный сотрудник,*
Красильникъянц Евгений Валерьевич *– канд. техн. наук, ст. научный сотрудник*
Ивановского государственного энергетического университета им. В.И. Ленина.
Контактный телефон (4932)26-97-52.
E-mail: krev@ispu.ru

Автоматизация получения идентификационных моделей дефектов поверхности прямошовных сварных изделий

Д.А. Проскурин, М.В. Овечкин, Л.В. Галина, В.Н. Шерстобитова (Оренбургский государственный университет)

Предложена методика получения агрегированных моделей для сигналов вихретокового дефектоскопа при неразрушающем контроле сварных прямошовных труб в процессе их изготовления. Приведено описание эксперимента, проводившегося на вихретоковом дефектоскопе. В качестве примера рассмотрены физически реализуемые идентификационные модели преобразования дефектов в выходной сигнал дефектоскопа. Результаты прошли апробацию на предприятиях: ЗАО «Друза» и ПК «УфаПромГаз» г. Оренбурга.

Ключевые слова: неразрушающий контроль, дефектоскопия, идентификационная модель, прямошовная сварка.

Согласно ГОСТ 10704-91, к прямошовным сварным изделиям из цельного металла, имеющим круглое или прямоугольное поперечное сечение, относят электросварные трубы.

Трубная отрасль России, традиционно сильная, в настоящее время восстанавливает свои позиции. Одной из проблем производства электросварных труб являются потери, обусловленные большими объемами брака трубной продукции. Решение задачи снижения потерь неразрывно связано с разработкой и внедрением новых способов и средств автоматизации производства и, в частности, автоматизации контроля качества металлоизделий, позволяющих в процессе изготовления изделий осуществлять их дефектацию и разделение на группы по пригодности к эксплуатации в различных условиях [1].

Теоретические основы автоматизированного контроля металлоизделий методами электромагнитной дефектоскопии заложены в трудах А.Л. Дорофеева, Р.Е. Ершова, В.Г. Герасимова, В.В. Клюева, Ю.Я. Останина, Г.С. Самойловича. Благодаря их работам стало возможным в значительной степени автоматизировать процесс контроля качества трубной продукции. Однако существующие методы разбраковки не обладают необходимой универсальностью и требуемой точностью.

Оценка качества трубной продукции осуществляется в основном за счет выявления дефектов типа нарушения сплошности или однородности металла. Сварные трубы в отличие от цельнотянутых имеют околошовную зону шва. Поэтому выделение и распознавание полезного сигнала на уровне шумов датчика дефектоскопа о состоянии поверхности трубы представляет собой сложную задачу, до настоящего времени не решенную.

К технологическому процессу производства сварных прямошовных труб предъявляются следующие технические требования:

— значительная скорость движения трубы в зоне контроля — до 3 м/с [2];

— использование одного датчика для контроля всего тела труб, включая сварной шов, непосредственно в процессе изготовления трубы;

— возможность быстрой переналадки дефектоскопа на разные диаметры труб (в диапазоне 32...89 мм) [2] и толщины стенок (в диапазоне 2...4 мм) [2].