Принципы построения программного обеспечения систем управления движением

А.П. Бурков, Е.В. Красильникъянц (ИГЭУ)

Рассмотрены вопросы, связанные с концепцией разработки ПО систем управления движением (СУД), предназначенных для автоматизации технологических установок. Отмечены возможные способы аппаратной и программной реализации таких систем. Предложена концепция построения СУД на основе автоматной реализации программ логического управления (PLC) и управления движением (PMC) посредством виртуальной машины. Приводится пример реализации автоматного программирования PLC программ на языке Forth.

Ключевые слова: система управления движением, программное обеспечение, виртуальная машина, автоматное описание, язык программирования.

Введение

Развитие современных технологий промышленного производства предъявляет ряд новых требований к средствам реализации задач управления. Традиционные методы построения систем управления во многих случаях не предоставляют достаточных возможностей для передовых систем с заданным уровнем точности и новыми возможностями. Возрастающая сложность новых производственных машин и агрегатов требует большей степени интеграции логических и технологических функций, а также реализации новых функций управления движением.

В связи с этим ряд иностранных фирм, таких как Siemens, Baldor Electric, Bosch Rexroth, Delta Tau, Rockwell Automation, Beckhoff Automation, Danaher Motion и др. приступили к разработке и выпуску нового поколения специализированных контроллеров, предназначенных для построения СУД. Областью применения этих контроллеров является создание систем управления, начиная от простых машин и агрегатов до сложных станков и производственных линий: упаковочные, текстильные и печатные машины; металлорежущие станки; робототехнические комплексы; машины, обрабатывающие пластмассы и резину; прессы, волочильные машины; крановая техника; обрабатывающие машины для дерева, стекла, керамики и камня. В зависимости от требований задачи предусматривается возможность реализации различных вариантов архитектуры системы управления, а также применение широкой номенклатуры компонентов, таких как двигатели, датчики, силовые преобразователи, контроллеры и т. д.

К отличительным особенностям СУД относятся требования высокого быстродействия и точности производимых вычислений, широкий спектр функциональных возможностей, специфичный набор измерительных и исполнительных устройств, развитый состав системного и прикладного ПО. Все виды СУД имеют в своем составе программные средства, обеспечивающие создание как программ логического управления, типичных для ПЛК, так и программ прямого цифрового управления электроприводами.

Как правило, ПО СУД должно иметь следующие свойства: расширенные функции ввода/вывода с обработкой и контролем ошибок; реализация многозадачности; задание приоритета исполняемых задач; коммуникация между задачами; переключение задач; управление ресурсами; возможность контролировать процессы циклически или по условию; возможность показывать статус процесса при контроле; обработка прерываний.

Общие принципы построения СУД

Открытая архитектура

Открытой считается архитектура, на которую опубликованы спецификации, что позволяет другим производителям разрабатывать дополнительные устройства для совместного использования.

Принцип открытой архитектуры предусматривает:

- возможность системной интеграции аппаратных и программных средств;
- базовое конфигурирование системы у производителя и окончательное – у пользователя;
- эволюцию системы в условиях минимальной зависимости от изменений системной платформы.

Актуальность использования принципа открытой архитектуры определяется тем, что при построении многих СУД состав функциональных блоков, предлагаемых изготовителем, не исчерпывает всех возможных вариантов построения системы. В аппаратной части это касается случаев, когда возникает необходимость использования различных типов двигателей, измерительных устройств и датчиков, интерфейсов связи. В состав СУД могут быть включены нестандартные или нестандартизированные элементы, например, системы технического зрения. Структура программных средств должна предоставлять возможность пользователям создавать и применять в СУД программные и аппаратные модули собственной разработки, предназначенные для решения специальных задач, без нарушения функциональной целостности системы.

В настоящее время разрабатываются стандарты на программное и аппаратное обеспечение модульных архитектур для контроллеров: OMAC (Open Modular Architecture Controls), OSACA (European Open System Architecture for Controls within Automation Systems,) OSEC (Japan Open System Environment for Controller Architecture).

Цифровая реализация

Принцип цифровой реализации подразумевает, что все основные компоненты СУ имеют цифровое исполнение и для их связи применяются высокоскоростные цифровые интерфейсы передачи данных. Такой подход позволяет:

- добиться масштабируемости и гибкости системы за счет возможности программной активации, настройки и управления подключаемыми модулями системы;

- обеспечить необходимую точность функционирования системы;

- избежать временных и разрядных потерь при цифро-аналоговых и аналого-цифровых преобразованиях;

- получить программный доступ и работу с удаленными компонентами, как с интегрированными в систему.

Важнейшим качественным показателем цифровой СУД является возможность выполнения сложных расчетов за заданный такт квантования по времени. В наиболее современных быстродействующих системах величина такта квантования находится в пределах 50...400 мкс. При управлении многоосевыми системами это предъявляет повышенные требования к быстродействию цифровых интерфейсов. Другой стороной перехода к малым тактам квантования во времени является неизбежное повышение разрядности обрабатываемых данных до 32 и 64 бит.

Модульность построения

Основными достоинствами модульной концепции являются возможность управлять стоимостью создаваемого продукта путем включения/исключения из него тех или иных функциональных модулей, простота установки и монтажа.

Модульное ПО включает группу файлов, каждый из которых допускает раздельную компиляцию. Такой подход обеспечивает значительное уменьшение затрат времени при изменениях, вносимых лишь в небольшое число исходных файлов, и упрощает групповую разработку. При этом появляется возможность замены отдельных компонент, не требующая переработки всего проекта.

Роль модулей могут играть структуры данных, библиотеки функций, классы, сервисы и др. программные единицы, реализующие заданный набор функций и предоставляющие интерфейс к ним. Наиболее высокую степень модульности обеспечивает объектно-ориентированное программирование (ООП) благодаря таким свойствам, как инкапсуляция, полиморфизм и позднее связывание.

Структура СУД представляет собой совокупность базовых и дополнительных модулей. Каждый модуль автономен, является вложенным объектом и располагает собственными: алгоритмической структурой, структурой данных и интерфейсной оболочкой. Взаимодействие модулей осуществляется посредством программной проблемно-ориентированной магистрали, которая не только поддерживает коммуникационные протоколы, но и выполняет управляющие функции.

Наличие модулей позволяет легче адаптировать систему к различным объектам управления. Модульный принцип построения допускает независимую разработку отдельных модулей системы, постепенно улучшая ее характеристики.

Реализация СУД на PC-совместимых компьютерах

Одним из развиваемых в настоящее время направлений при создании СУД является использование PC-совместимых компьютеров (PC-base) на основе ОС РВ [1]. Несмотря на высокую вычислительную мощность процессоров, реализовать этот подход в полной степени для СУД не удается вследствие ряда причин [2].

Поскольку в однопроцессорных системах непосредственные параллельные вычисления невозможны, необходим механизм, обеспечивающий квазипараллельную обработку процессов, – ОС РВ.

В системе РВ используется таймер, формирующий определенную частоту прерываний, интервал между которыми называется квантом времени. Каждое прерывание запускает программу-обработчик, увеличивает локальный счетчик на единицу и проверяет, нет ли в очереди задачи, которую следует перевести в состояние готовности и не пора ли провести переключение задач.

Каждая задача содержит блок управления задачей – timer control bloc (TCB), код программы и собственную область данных для переменных. В TCB имеется указатель на начало программы, адрес области данных, статус задачи, ее приоритет и область сохранения регистров. Если задаче предоставлен квант времени программой-диспетчером, то в этом случае контекст предшествующей задачи должен быть сохранен в оперативной памяти, а контекст новой задачи устанавливается для исполнения.

Недостатками ОС РВ можно считать высокую стоимость ПО, обусловленную широким набором предоставляемых сервисов, отсутствие программных средств для работы с нестандартными высокоскоростными интерфейсами, значительные затраты времени на переключение контекста задач программой-диспетчером. Реализация задач, типичных для СУД, в рамках ОС РВ приводит к зависимости программ от ОС, что снижает мобильность разрабатываемого ПО. С этой точки зрения более предпочтительным следует считать разработку специализированных микро-ОС, учитывая, что программы управления движением имеют гораздо более ограниченный круг задач по сравнению с задачами, решаемыми на ПК.

Реализация СУД на контроллерах движения

Другим вариантом построения СУД, получившим распространение в последнее время, является использование контроллеров движения (КД). Это высокопроизводительный автономный компьютер, способный выполнять системное и прикладное ПО для отработки процессов и событий в РВ, реализуя многокоординатное управление перемещением исполнительных органов. Основой КД является процессорный модуль, содержащий высокопроизводительный процессор, оперативную память, энергонезависимую память для хранения системного ПО, энергонезависимую память для хранения программ пользователя, а также сторожевой таймер, необходи-

мый для предотвращения аварийных ситуаций. Для связи с внешними по отношению к процессорному модулю компонентами КД, как правило, используется шина данных процессора.

Контроллер движения поставляется с обычными средствами разработки типа VB, C/C++ или с проблемно-ориентированными языками программирования собственной разработки фирм-производителей. Компиляция управляющих программ может происходить либо на хост-компьютере с последующей загрузкой во Flash-память, либо непосредственно на КД. Кроме того, для отладки СУД предлагаются программные средства, позволяющие вести диалоговый обмен, сбор данных и их визуализацию, мониторинг текущих процессов, автоматизировать процесс настройки приводов.

В КД интегрированы программные средства для эффективной реализации следующих основных задач СУ: вычисление и формирование траектории движения, расчет разгонов и торможений электродвигателей; формирование и отработка задающих воздействий в контурах регулирования током, скоростью и положением; выполнение логических операций по управлению в реальном и машинном времени фоновыми алгоритмами и цикловыми механизмами электроавтоматики. В КД обычно нет самостоятельной ОС, и изготовители поставляют те или иные варианты ПО разработки СУД.

Применение специализированных контроллеров СУД позволяет реализовать более широкий спектр возможностей взаимодействия с нестандартизированным оборудованием, таким как системы технического зрения, информационными и измерительными комплексами. Однако в этой связи возникает необходимость создания средств разработки ПО, специализированного для использования в СУД.

Использование виртуальной машины для разработки СУД

Опыт разработки и эксплуатации СУД показывает, что в настоящее время период жизни большинства микропроцессорных платформ довольно краток и обычно не превышает 10 лет. Исключением можно считать архитектуру Intel x86, которая по ряду причин не является оптимальной для построения СУД. В этой связи важное значение имеет сохранение преемственности для разрабатываемых программ управления и систем разработки. Не менее важно обеспечить полное и ясное документирование управляющих программ, доступное для широкого круга программистов и системных интеграторов.

Обеспечение мобильности ПО для встроенных систем требует использования языка программирования, обеспечивающего поддержку специфических для конкретных применений аппаратных особенностей, возможность использования виртуальной ОС и полной открытости для программиста. Для преодоления структурных различий между системами команд может быть использована простая программная вычислительная машина, имеющая стандартный набор регистров и выполняемых операций. Все программы пользователя описываются в терминах операций этой виртуальной машины (ВМ). При смене аппаратной платформы необходимо изменить лишь ассемблерную реализацию этой машины, не изменяя системные программы и программы пользователя.

Используемый язык программирования должен обеспечивать модульное построение аппаратного и программного обеспечения, которое дает следующие преимущества: снижение затрат на разработку ПО; раздельное тестирование отдельных модулей; возможность изменяемости и расширяемости системы.

Для систем разработки СУД желательно выполнение следующих требований: расширяемое стандартное ядро (открытая архитектура); контекстно-зависимый поиск; векторизованное исполнение программ; русскоязычный интерфейс; поэтапная отладка в РВ.

Структура ПО контроллера СУД

Для ускорения разработки целесообразно иметь стандартное ядро минимального объема с возможностью расширения или изменения любых функций силами разработчика прикладного ПО. В связи с этим ядро системы и все вышестоящие программы следует реализовывать с помощью промежуточного кода, исполнение которого зависит от программной модели ВМ, реализованной для данного процессора (рис. 1).

На нижнем уровне системы, кроме ВМ, реализуются программы, требующие наивысшего быстродействия, такие как цифровые регуляторы, широтно-импульсные модуляторы и драйверы интерфейсов, зависимые от используемой аппаратной платформы. На следующем уровне, соответствующем функциям микро-ОС, производится управление выполнением всех программ. Используемые соглашения определяют порядок взаимодействия всех остальных уровней между собой. На среднем уровне находятся служебные программы, используемые главным образом на этапе загрузки и отладки PMC и PLC программ: компилятор и редактор. На уровне служебного интерфейса определяются виды и способы взаимодействия с хост-компьютером через доступные каналы связи. На верхнем, пользовательском уровне расположены библиотечные функции PLC и PMC программ. Расширения на этом уровне могут производиться пользователем системы.

Основное достоинство данного подхода заключается в том, что не существует принципиального различия между уровнями. Операционная оболочка, компилятор и редактор, представляющие собой совокупность загружаемых компонент, построены по модульному принципу тем же способом, что и программа пользователя, и доступны для любой необходимой модификации. Кроме того, отсутствует необходимость в кросс-компиляторе в составе системы разработки.

Отлаженные программные модули, загружаемые в текстовом формате через внешний интерфейс, ком-

Рис. 1. Структура ПО СУД, основанная на виртуальной машине

пилируются встроенным компилятором, временно располагаемым в свободной оперативной памяти. Каждый модуль должен содержать самостоятельный программный компонент, обеспечивающий выполнение законченного набора операций над определенными объектами, и собственную секцию самоинициализации. Настройка системы для выполнения конкретных задач должна производиться путем загрузки необходимого набора сервисных и пользовательских программных модулей.

Разделение времени в СУД

Программы прямого цифрового управления (PMC) должны вызываться через фиксированный интервал времени, значение которого лежит в пределах от единиц до сотен микросекунд. Эти программы имеют безусловный приоритет перед любыми другими, поскольку нарушение работы этого условия может вести к непредсказуемым последствиям. Некоторые из программ логического управления (PLC) также могут требовать фиксированных интервалов запуска (обычно в пределах единиц мс). Основная часть PLC программ работает с относительно медленными источниками и приемниками информации, такими как реле, контакторы, клавиатуры, гидро- и пневмоклапаны, которые некритичны к изменениям временных задержек в пределах 0...50 мс. С другой стороны, обработка PLC программ на современных микропроцессорах занимает очень мало времени, сводясь к выполнению нескольких машинных команд. В связи с этим появились упрощенные варианты микро-ОС для СУД, в которых переключение задач логического управления по тайм-

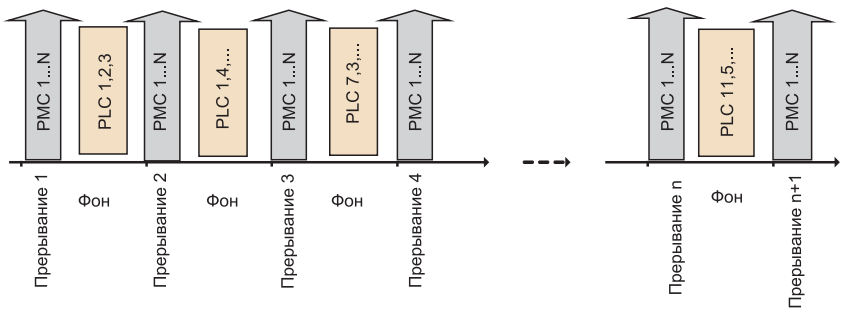

Рис. 2. Разделение времени между PMC и PLC программами

меру не производится. Поскольку из PLC программ устраняются операторы цикла, сами программы распадаются на короткие фрагменты. При этом происходит значительное упрощение операционной части, снимается возможность образования критических секций задач и нарушения целостности данных. Обмен данными между PLC и PMC программами производится с помощью "атомарных" команд. Типичная структура временных диаграмм обработки в подобной системе приведена на рис. 2.

Из рисунка видно, что поток команд разделяется на два временных слоя: синхронный для PMC и части PLC программ и фоновый асинхронный для большинства PLC программ. Разбор командной строки, ввод текста, компиляция и отладка программ происходят в фоновом режиме, без нарушения основного потока команд.

Обработка ошибок в СУД

Особые требования предъявляются к анализу и обработке исключительных ситуаций в СУД, так как последние используются в ответственных и потенциально опасных машинах. Во встраиваемых системах управления недопустимо, чтобы система "зависала" или останавливалась при возникновении ошибок. Для предотвращения таких ситуаций необходимо иметь возможность определить обработчик ошибок – программу exception, которая выполняется в случае возникновения предусмотренных ошибок. Этой программе должны передаваться параметры, определяющие код источника и номер ошибки.

Автоматное описание объектов СУД

Для повышения надежности ПО СУД и снижения вероятности появления ошибок целесообразно использовать для построения программ достаточно крупные модули – объекты, представляющие собой объединение локальных областей данных и связанных с ними алгоритмов. С этой целью можно определить типовые программные модули, характерные для СУД, и разработать для них необходимые схемы компиляции.

Наиболее распространенными видами отображения логических схем являются граф-схемы алгоритмов, таблицы переходов и выходов, а также графы автоматов [3]. Обычно автоматами называют некоторые алгоритмы работы двоичных систем логики. Однако понятие автомата оказалось удобным и для разработки компьютерных программ (CASE или SWITCH технология). Аналогичные методы "диаграмм состояний" используются в программном комплексе Matlab. В работах [4, 5] приводятся примеры разработки построения конечных автоматов (КА) для программирования компьютеров. Некоторым недостатком можно считать отсутствие в большинстве компиляторов соответствующих управляющих конструкций, так как использование операторов CASE или SWITCH для реализации автоматов достаточно не-

удобно. Между тем, ряд иностранных фирм располагает соответствующим ПО и уже использует в своих разработках автоматное представление управляющих алгоритмов, например, фирма Siemens в электроприводах SINAMICS.

Все текущие состояния объектов логического управления (реле, кнопок, пускателей, регуляторов и задатчиков) определяются состояниями набора конечных автоматов. Под состоянием автомата понимается подпрограмма, подлежащая циклическому выполнению в процессе работы (регуляторы, задатчики, таймеры и др.). В случае выполнения некоторых логических условий возможны переходы от одного состояния в другое. При переходе могут выполняться однократные действия (например, сообщения об изменении режима работы). В случае необходимости фиксировать лишь факт изменения (например, нажатие кнопки) состояния могут не производить никаких вычислений, но при переходе из одного состояния в другое возможно выполнение однократных действий. Любое состояние само может быть автоматом, активизируемым при выполнении данного состояния.

При использовании данной технологии разработка управляющей программы начинается с создания графического образа управляющей программы или процесса – направленного графа с числом вершин, равным числу возможных состояний. Каждое состояние определяет программу или процесс, который должен выполняться при передаче управления автомату. Определяются условия перехода от одного состояния к другому – события. В случае необходимости инициализации следующего состояния во время перехода определяются действия по инициализации. На основе направленного графа по типовому шаблону или автоматически может быть составлен текстовый файл для последующей компиляции.

Автоматы могут быть использованы для построения широкого класса объектов управления в СУД. Например, если разгон и торможение электропривода производятся регулятором с прямой связью, а работа на постоянной скорости должна происходить с прямой и обратной связью, создается автомат, состояниями которого может быть обозначена работа с прямой и комбинированной связью.

Условием перехода от состояния 1 к состоянию 2 будет являться событие, заключающееся в достижении заданной скорости. Условием обратного перехода может быть событие, заключающееся в получении команды останова. Очевидно, что при переходе от состояния 1 к состоянию 2 необходимо установить в 0 значение интегральной части регулятора, если она имеется. Данный автомат должен быть установлен в один из синхронных потоков управления. Аналогичным образом могут быть построены PLC программы управления, аварийной блокировки, задания режимов работы и т.д.

Задачей программ управления движением (PMC), например, JOG или S-кривой, является формирование временных диаграмм заданий положения или скорости привода. В данном случае каждое состояние автомата, формирующее задание по одному из определенных законов, должно быть выполнено заданное число раз. Условием перехода в следующее состояние является реализация заданного числа операций, выполняемых в текущем состоянии.

Для обеспечения отладки и контроля за текущим состоянием СУД значения заданного числа изменений всех состояний записываются в журнал, расположенный в оперативной памяти, что позволяет вести мониторинг состояния системы, фиксировать последовательность развития аварийных ситуаций. Возможно также ситуационное управление, когда анализ последовательных переходов из состояния в состояние формирует новое событие, передающее управление новому автомату.

Информация о состоянии всех автоматов может передаваться по высокоскоростному интерфейсу в хост-компьютер и отображаться на экране монитора. В сложных системах для исключения возможности создания нестабильных или безвыходных состояний возможно создание тестовых генераторов событий с последующим анализом поведения СУД.

Другим часто используемым программным компонентом являются кольцевые буферы – участки оперативной памяти, предназначенные для временного хранения данных. С буфером связаны два указателя и две программы записи и чтения из кольцевого буфера. Кроме того, с буфером связан логический автомат, контролирующий заполнение и опустошение буфера. При наступлении соответствующих событий автомат может подавать управляющие сигналы интерфейсам ввода/вывода.

Кольцевые буферы могут являться составной частью генераторов траекторий (ГТ) и координатных систем (КС). ГТ представляют собой программу, заполняющую кольцевой буфер значениями задания положений для одного или нескольких двигателей, соответствующими некоторому отрезку траектории движения. КС включают в состав кольцевые буферы интерфейса с хост-компьютером, интерпретаторы команд, набор различных ГТ, соответствующих набору командного интерфейса.

Реализация автоматных описаний на язые Forth

Возможности использования автоматов могут быть обеспечены при использовании языка высокого уровня Forth, разработанного в начале 70-х годов XX века Чарльзом Муром на основе оригинальной концепции двухстековой ВМ. Наиболее характерными чертами языка являются предельная простота, максимальная гибкость и расширяемость в произвольном направлении. В настоящее время язык широко применяется для разработки ПО встраиваемых микропроцессорных систем различного назначения.

Свойства модульности, открытость внутренней структуры, предельная компактность и высокая мобильность обеспечили широкое использование языка

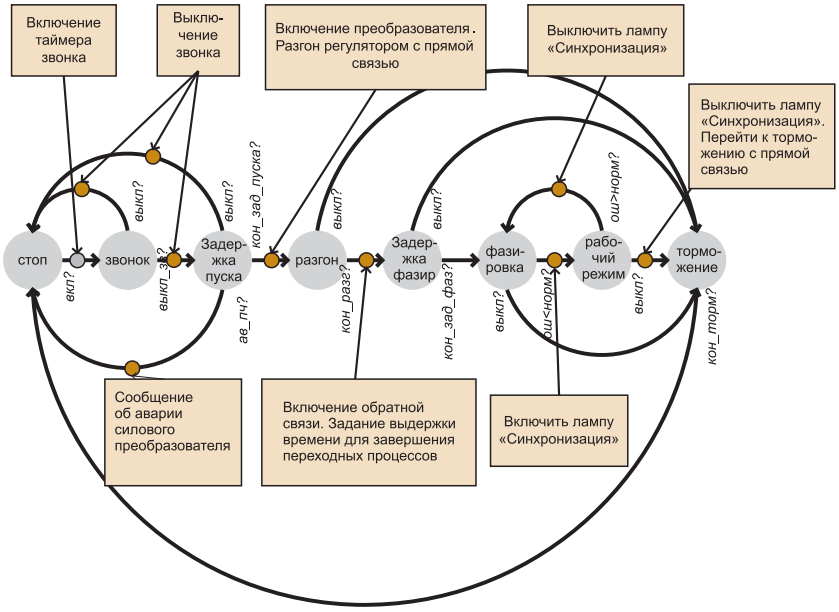

Рис. 3. Направленный граф переходов и состояний автомата логического управления электроприводом стеклоформующей машины

в системах РВ. В Forth возможно использование любых средств языка на любом этапе компиляции или исполнения. Язык характеризуется: простотой и компактностью реализации; максимальной расширяемостью и гибкостью; полной открытостью внутренней структуры; возможностью создания новых типов данных, управляющих конструкций и определяющих слов; простотой использования в комбинации с ассемблером.

В основной программе организуется цикл выполнения, в котором имена автоматов используются подобно обычным Forth-словам. В процессе разработки программ интерпретатор текста последовательно компилирует исполнимые токены команд Forth -машины.

Определяющее слово fsm (используется в контексте n fsm NAME_fsm) при вызове снимает со стека число состояний создаваемого автомата и создает структуру управления, определенного вида. Для каждого автомата в области ОЗУ выделяется битовое поле, максимальное числовое значение которого соответствует числу состояний данного автомата. Адрес битового поля расположен в информационном слове.

```
8 fsm режимы        \ определить автомат "режимы" на 8 состояний
states режимы       стоп  звонок  задержка_пуска  разгон
задержка_фазир фазировка рабочий_режим торможение \ определить
состояния автомата "режимы"
\ определить переходы в другие состояния
jmp-at стоп         вкл?        to звонок          вкл_зв
end-jmp
jmp-at звонок       выкл_зв?    to задержка_пуска  выкл_зв
                    or выкл?    to стоп            выкл_зв
end-jmp
jmp-at
задержка_пуска      кон_зад_п?  to разгон          включи
                    or выкл?    to стоп            выключи
                    or ав_пч?   to стоп            выключи
end-jmp
jmp-at разгон       кон_разг?   to задержка_фазир
                    or выкл?    to торм
end-jmp
jmp-at
задержка_фазир      фаз_разр?   to фазировка
                    or выкл?    to торм
end-jmp
jmp-at фазировка    ош<норм?    to рабочий_режим   вкл_лампу
                    or выкл?    to торм
end-jmp
jmp-at
рабочий_режим       ош>норм?    to фазировка       выкл_лампу
                    or выкл?    to торм            выкл_лампу
end-jmp
jmp-at торм         кон_торм?   to стоп            выключи
end-jmp
end-fsm
```

Рис. 4. Программный код работы автомата логического управления

Ниже представлено текстовое описание и графическое отображение структуры одного из 24 автоматов логического управления, разработанных для электропривода стеклоформующей линии [6]. Разработка автомата начинается с построения направленного графа, отображающего необходимую последовательность режимов и переходов (рис.3).

Однократные действия на диаграмме состояний представлены в виде окружностей, расположенных на дугах, каждая из которых соответствует переходу в новые состояния (на выносках приводится содержание этих действий). Переходы осуществляются в том случае, если проверка условий переходов, постоянно выполняемая в каждом состоянии, возвращает логическое значение true. Если условия переходов не являются взаимоисключающими, приоритет выполнения имеет условие, записанное в тексте первым.

По рисунку графа можно составить текст исходной программы (рис. 4) (подразумевается, что содержание слов – состояний, переходов и действий определено выше по тексту).

На рисунке и в программе зеленым цветом выделены слова, соответствующие возможным состояниям, красным цветом отмечены однократные действия, выполняемые при переходе, а курсивом – условия переходов.

Для начала использования автоматов в контроллер должен быть загружен программный компонент fsms, обеспечивающий компиляцию логических автоматов. При вводе определяющее слово fsm совместно со словом states создает управляющую часть, резервирует битовое поле для информации о текущем состоянии, формирует заготовку таблицы векторов переходов для состояний и секцию инициализации. Затем с помощью определяющих слов jmp-at, or, end-jmp и условий переходов компилируется основной алгоритм автома-

та. Завершающее слово end-fsm проверяет правильность использования всей конструкции.

После ввода текстового файла в контроллер ПО хост-компьютера копирует полученный объектный код в собственную оперативную память и по структуре кода восстанавливает графический образ автомата. После этого код автомата может быть загружен в фоновый поток управления. Проверка функционирования производится с помощью подсистемы графического отображения автоматов.

При запуске системы управления начальным состоянием является "стоп", которое не выполняет никаких действий, однако всегда анализирует условия переходов. Если была нажата кнопка "пуск", что проверяется условием "вкл?", производится переключение в состояние "звонок" и включение звонка предупредительной сигнализации словом "вкл_зв" и выход из автомата. Исполнение состояния "звонок" и проверка условия выхода из него будут производиться после того, как автомат "режимы" снова получит управление.

ПО хост-компьютера, работающего с контроллером СУД, построенным на основе ВМ, практически не зависит от ПО контроллера. Минимальный набор функций, необходимых для редактирования, загрузки и отладки ПО, может быть обеспечен набором стандартных сервисов ОС. Однако более предпочтительным является дополнение подсистемой графических средств отображения структуры и текущего состояния автоматов, что существенно облегчает и ускоряет процесс отладки и ввода в эксплуатацию. В некоторых случаях, например, при использовании в станочном оборудовании хост-компьютер может служить и источником управляющей информации для контроллера.

Выводы

Рассмотрены особенности создания ПО контроллера в составе СУД на основе предложенного варианта построения ВМ. Послойная организация структуры ПО СУД обеспечивает простое расширение и дополнение системы разработки при достижении высокой мобильности программ. Автоматная реализация объектов СУД позволяет поддерживать простое документирование управляющих программ. Использование языка Forth позволяет обеспечивать наглядную и высокоэффективную разработку и отладку специализированного ПО.

Представленные результаты были успешно использованы при разработке СУД в установках для производства полимерного оптического волокна (ВНИИСВ), линии радиационно-химической отделки линолеума (завод "ИСкож"), серии роторных стеклоформующих агрегатов (ВВ7,U8,U12,S10) [6]. ПО логических автоматов, первоначально разработанное для 16-битной аппаратной платформы MCS196 Intel, было практически без изменений перенесено на 32-битную платформу ARM7. Предлагаемые принципы построения ПО СУД возможно использовать и с другими языками программирования при условии соответствующей доработки компиляторов.

Список литературы

1. *Захаров Н.А.* ПЛК и PC-совместимые контроллеры: два подхода к построению систем // Автоматизация в промышленности. №4. 2003.
2. *Бурков А.П., Красильникъянц Е.В.* Принципы построения контроллеров движения // Тр. международной научно-технической конференции по автоматизированному электроприводу "АЭП 2007". С.-Петербург. 2007.
3. *Баранов С.И.* Синтез микропрограммных автоматов. Л.: Энергия. 1979.
4. *Шалыто А.А.* SWITCH технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998.
5. *Шалыто А.А.* Использование граф-схем и графов переходов при программной реализации алгоритмов логического управления //Автоматика и телемеханика. 1996. № 6,7.
6. *Красильникъянц Е.В.* Новые системы управления для стеклоформующих машин // Стеклянная тара. 2003. №5.

Бурков Александр Павлович *– канд. техн. наук, ст. научный сотрудник,*
Красильникъянц Евгений Валерьевич *– канд. техн. наук, ст. научный сотрудник кафедры "Электроники и микропроцессорных систем" Ивановского государственного энергетического университета.*
Контактный телефон (4932) 26-97-52.
E-mail: krev@ispu.ru Http://www.inelsy.com

Интеграция сетей EtherNet/IP и CANopen с помощью Anybus X-gateway CANopen

Компания HMS Industrial Networks представляет новый шлюз Anybus X-gateway CANopen, позволяющий обеспечивать передачу данных ввода/вывода между автоматическими устройствами в сетях EtherNet/IP и CANopen. Конфигурируемый и плоский автономный шлюз монтируется на стандартную DIN-рейку и запитывается от промышленного источника питания напряжением 24 В. Он функционирует как адаптер (ведомый) в сети EtherNet/IP и в качестве устройства управления (ведущий) на стороне CANopen. Функция ведущего CANopen конфигурируется с помощью гибкого и простого в употреблении конфигурационного инструмента, работающего в ОС Windows, прилагаемого к изделию. В шлюзе на стороне EtherNet/IP имеется встроенный двухпортовый коммутатор, позволяющий устанавливать EtherNet/IP в шинной или линейной топологии и обходиться без установки внешних коммутаторов.

Таким образом, семейство изделий Anybus X-gateway CANopen поддерживает интеграцию и связь до 10 различных промышленных шин и сетей Ethernet, таких как Profibus, DeviceNet, CANopen, EtherCAT, Modbus и Profinet. Все версии работают одинаково, делая устройство наиболее гибким стандартным решением на рынке для соединения двух промышленных сетей.

Http://www.anybus.com и www.industrialnets.ru

Http://www.avtprom.ru