Козаченко В.Ф.

Эффективный метод программной реализации дискретных управляющих автоматов во встроенных системах управления

В статье предлагается новый метод программной реализации дискретных управляющих автоматов для систем встроенного управления реального времени, таких как системы прямого цифрового управления силовыми преобразователями и приводами с двигателями различных типов. Метод позволяет эффективно решать задачи управления как собственно силовым преобразователем, так и дополнительным релейно-контакторным оборудованием, например, оборудованием групповых насосных станций. Он основан на теории автоматов и ее приложениях, развивающих автоматный подход к разработке программного продукта. Основным достоинством метода является технология непосредственной реализации программы управляющего автомата по графу переходов, который создается в строгом соответствии с техническим заданием. Метод обеспечивает точное документирование разработки, существенно облегчает отладку и сопровождение программного продукта. Предназначен для использования в системах автоматизации, энерго- и ресурсосбережения, пригоден для систем распределенного сетевого управления. Реализован в отечественной серии преобразователей частоты «Универсал», поддерживающих групповое управление много-насосными агрегатами.

1. Классификация задач управления современными силовыми преобразователями

Современная преобразовательная техника строится на основе интеллектуальных силовых модулей и специализированных микроконтроллерных систем управления, адаптированных по архитектуре, производительности и набору интегрированных на кристалл периферийных устройств к задачам прямого цифрового управления оборудованием, вплоть до формирования высокочастотных широтно-импульсных сигналов управления (ШИМ) всеми силовыми ключами. При этом специализированная периферия (аналого-цифровые преобразователи, менеджеры событий, квадратурные декодеры и т.д.) работают практически автономно, отвлекая центральный процессор по прерываниям только для получения нового задания или возврата результата (например, результата серии последовательных аналого-цифровых преобразований). Общее число запросов прерываний в подобных системах может достигать нескольких десятков.

Характерной особенностью таких систем является необходимость жесткой иерархии решения множества задач с различной частотой (интервалом квантования по времени). Так, задачи прямого цифрового управления оборудованием (задачи нижнего уровня), в частности задачи управления силовым ключами преобразователя, должны решаться на каждом периоде ШИМ (при частоте несущей 10-20 кГц за время 50-100 мкс). Например, в преобразователях частоты (ПЧ) со скалярной системой управления асинхронными двигателями за указанное время должен выполняться расчет текущей фазы вектора напряжения статора, должны определяться и задаваться уставки на сравнение для всех каналов ШИМ-генератора. В ПЧ с векторной системой управления за то же время должны решаться задачи регулирования токов, преобразования координат, определения электрических и механических координат привода и ряд других.

Задачи среднего уровня управления могут решаться с существенно меньшим интервалом квантования по времени (единицы, доли мс): формирование требуемых тахограмм разгона-торможения (задатчики интенсивности), регулирование скорости и

технологических переменных (технологические регуляторы) и им подобные. Интервал квантования для этого уровня управления может задаваться либо от отдельного интервального таймера, либо кратным числом периодов ШИМ.

На верхнем уровне решаются задачи управления режимами работы преобразователя (интерпретатор команд оперативного управления преобразователем), интерфейса с оператором и с системами управления более высокого уровня. Здесь же решаются задачи согласованного управления релейно-контакторным оборудованием рабочих станций и режимами работы собственно преобразователя частоты, а также поддержки распределенного сетевого управления оборудованием.

Сложность указанных выше задач диктует применение в таких системах специализированных сигнальных микроконтроллеров повышенной производительности (до 20-40 млн. оп./с и выше) серии Motor Control [1, 2]. С одной стороны, эти микроконтроллеры имеют архитектуру центрального процессора, оптимизированную для предельно быстрого решения задач цифровой фильтрации и регулирования, а с другой – мощный набор встроенных периферийных устройств, обеспечивающих прямое цифровое управление оборудованием, вплоть до одновременного управления 16-ю силовыми ключами в режимах векторной синусоидальной ШИМ-модуляции.

Настоящая статья предлагает методику решения задач третьего уровня, пригодную как для управления режимами работы собственно ПЧ, так и для управления релейно-контакторными станциями группового управления оборудованием, в том числе по сетевым каналам связи.

2. Краткое введение в историю вопроса

Известны различные методы программной реализации дискретных управляющих автоматов, базирующиеся на эквивалентном аппаратном представлении автомата в виде трех устройств: входного логического преобразователя, памяти и выходного логического преобразователя. В такой модели автомата память обеспечивает запоминание текущего состояния автомата Q[k], выходной логический преобразователь – формирование вектора выходных управляющих воздействий Y[k]=Y(Q[k]), а входной логический преобразователь – формирование вектора перехода автомата в функции вектора полного состояния автомата Q[k+1]=f(X[k], Q[k]), т.е. текущего состояния автомата и текущих значений входных воздействий. В том случае, когда в качестве модели автомата принимается не автомат Мура, а автомат Мили или смешанный автомат (Мура-Мили), вектор выхода определяется как текущим состоянием автомата, так и текущим состоянием входов Y[k]=Y(X[k], Q[k]).

По существу программная реализация дискретного автомата сводится к реализации двух логических функций: функции перехода и функции выхода. При малой размерности векторов входа и состояния автомата задача может быть решена табличным способом. С увеличением размерности векторов входа и состояния автомата табличный способ становится неприемлемым и применяется метод непосредственного вычисления логических функций с использованием логических команд. Разновидностью метода является представление каждой логической функции в виде граф-схемы алгоритма с последующим использованием для его реализации команд тестирования битовых переменных. Два последних способа особенно эффективны при наличии битового процессора, что исключает необходимость распаковки и упаковки битовых переменных.

Казалось бы, что проблемы программной реализации дискретных автоматов не существует. На деле же, все упирается в один, очень непростой вопрос: как по техническому заданию, сформулированному технологом, перейти к функциям перехода и выхода автомата? Разумеется, при достаточной настойчивости разработчика, а особенно при условии декомпозиции задачи, когда она представлена иерархией взаимодействующих между собой автоматов, такие функции могут быть получены,

минимизированы и реализованы. Но, малейшие изменения алгоритма приведут к необходимости повторного решения задачи и к большим затратам на разработку.

Сложность проблемы отмечается всеми специалистами в области промышленной автоматики. Даже используя самый современный промышленный контроллер со специальным языком программирования, например, языком релейно-контакторных схем или булевой алгебры, вам придется прежде всего сформулировать задачу управления. Именно в этом будут проблемы, а не в программировании и кодировании.

Где же выход? За последние несколько лет ведущее производители промышленных программируемых контроллеров пытаются повернуться в сторону пользователей и предложить им методы интуитивно понятного программирования. Речь идет о формулировке задачи на языке циклограмм, сетей Петри, графов переходов автоматов. Так, версии языков программирования промышленных контроллеров на основе графов переходов автоматов предлагают фирмы «Шнайдер Электрик», «Сименс», «Фесто» и ряд других. Очевидно, что использование таких языков во встроенных системах управления невозможно, как из-за отсутствия трансляторов, так и из-за ограничений по быстродействию.

Рядом отечественных ученых А.А. Шалыто, Н.И. Туккель [3, 4] и другими в последние годы усиленно развивается так называемый автоматный подход к решению задач логического управления, когда и для формулировки задачи и для ее реализации предлагается использовать графы переходов взаимодействующих между собой автоматов. Разработанный нами метод близок к упомянутому выше и отличается от него главным образом, *технологией обработки событий, адаптированной к задачам быстродействующего прямого цифрового управления оборудованием в реальном времени*. Второе отличие заключается в *использовании входных переменных нескольких типов*, в том числе переменных управления внутренними процессами, конфигурационных переменных, способных изменять логику работы управляющего автомата, а также команд управления, поступающих по сетевым каналам связи или от пультов управления. Третье отличие связано с *расширением метода на распределенные системы управления*, когда отдельные автоматы или подавтоматы могут взаимодействовать между собой по локальным промышленным сетям, в частности по CAN-сети. Аппаратная поддержка метода обеспечивается разработанной серией специализированных контроллеров для управления двигателями и статическими преобразователям энергии, в состав которой дополнительно входят интеллектуальные платы ввода/вывода дискретных сигналов и пульты оперативного управления [5].

3. Основные положения метода

1. Любая задача логического управления, в том числе режимами работы преобразователя частоты, релейно-контакторным оборудованием насосной станции и т.д., исключительно на основе требований технического задания и опыта разработчика может быть сформулирована в терминах *ограниченного (счетного) числа возможных состояний*, в которых может находиться оборудование (режимов работы оборудования). Каждому из таких состояний оборудования поставим в соответствие состояние управляющей программы (дискретного управляющего автомата) на текущем такте (скане) работы автомата.
2. Последовательно, начиная с нулевого, пронумеруем все возможные состояния автомата и свяжем *текущее состояние автомата* с *переменной состояния автомата* Q[k]. При инициализации системы управления (при включении питания) переменную состояния автомата будем автоматически обнулять, считая начальное состояние нулевым. Если при отключениях питания требуется режим «самозапуска», то восстановление переменных состояния производится из энергонезависимой памяти.

3. Введем еще одну переменную Q[k-1], которая будет отображать состояние автомата при предыдущем вызове – *предыдущее состояние автомата* (тоже обнуляется при инициализации). Наличие двух переменных (текущего и предыдущего состояний автомата) позволит однозначно отличить первое вхождение в состояние от всех последующих вхождений:

   If Q[k]≠Q[k-1] Then {/*Первое вхождение*/ Q[k-1]=Q[k]}

   Else {/*Последующее вхождение*/}

4. Для переменных текущих и предыдущих состояний автоматов выберем *информативные имена*, например, ПЧ - состояние преобразователя частоты (режим работы), РС – состояние рабочей станции, Дв_Сеть – состояние аппаратуры, подключающей двигатель непосредственно к сети переменного тока. Договоримся, что переменная предыдущего состояния будет отличаться от переменной текущего состояния только суффиксом «_ OLD», например ПЧ_OLD. Будем отображать условие нахождения ПЧ в пятом состоянии логическим выражением ПЧ=5 или логической переменной ПЧ5 (ПЧ5=1 когда преобразователь находится в 5-м состоянии).

5. Выполним декомпозицию задачи управления и представим ее в виде *иерархии взаимодействующих между собой автоматов*. Возможны варианты (рис. 1):

   - *Централизованная система управления*: головной автомат управляет работой нескольких подчиненных автоматов (*подавтоматов*). Например, автомат управления рабочей станцией взаимодействует с несколькими подавтоматами: управления преобразователем частоты, управления подключением и отключением двигателя к/от сети, управления подключением и отключением двигателя к/от преобразователя частоты.

   - *Децентрализованная система управления*: параллельно работают несколько управляющих автоматов. Например, автомат управления групповой насосной станцией и автомат управления механической задвижкой.

   - *Сочетание централизованных и децентрализованных систем управления*: часть управляющих автоматов работает параллельно, часть - выполнят функции подавтоматов.

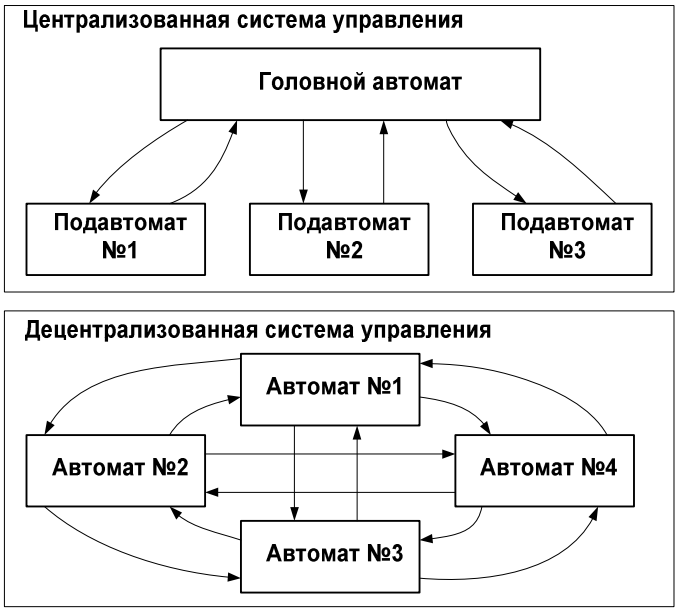

Рис. 1. Типовые структуры управления

6. Примем, что все параллельно работающие *автоматы могут взаимодействовать между собой с помощью переменных состояния*, доступных для окружения, т.е. *переменная состояния* одного автомата, являясь его выходом, может быть использована *в качестве входного воздействия* для другого автомата и наоборот. Автоматы обмениваются между собой сообщениями, меняя коды своих состояний. Например, условие нахождения рабочей станции в состоянии PC5 можно использовать в качестве входного воздействия в автомате управления режимами работы преобразователя частоты.
7. Использование переменных состояния позволяет *строго синхронизировать между собой различные процессы*. При этом переменная состояния головного автомата может быть использована для запуска подчиненного процесса, а переменная состояния подчиненного автомата – для передачи головному автомату информации о завершении подпроцесса. Пример на рис. 2. иллюстрирует: механизм запуска подчиненного процесса в состоянии A1 основного процесса; ожидание основным процессом завершения подчиненного процесса – переход в состояние A2 по условию достижения подпроцессом конечного состояния B3; механизм сброса подчиненного процесса в начальное состояние B0 при условии, что основной процесс идентифицировал факт завершения подпроцесса. Таким образом, *через переменные состояния обеспечивается* как *управление процессами*, так и *обратная связь о ходе процессов*.

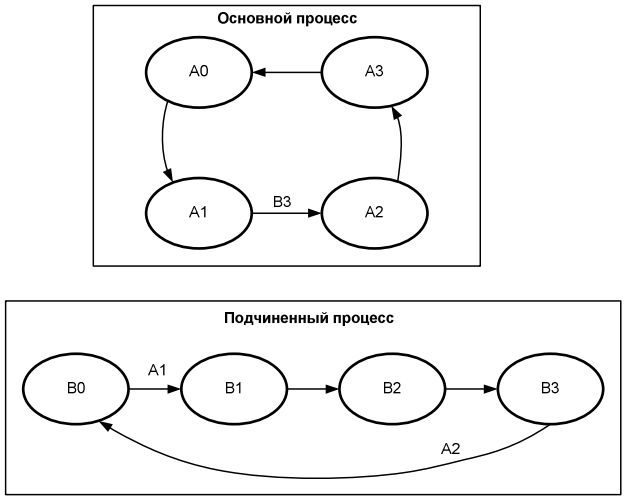

Рис. 2. Взаимодействие двух процессов

8. Допустим взаимодействие автоматов между собой не только за счет переменных состояния, но и за счет *внутренних выходных воздействий* - *битовых управляющих переменных*. Внутренние выходные воздействия формируются аналогично внешним воздействиям (выдаваемым во вне) в определенных состояниях автоматов. Например, автомат управления преобразователем частоты в качестве входных воздействий может обрабатывать битовые выходные воздействия «Пуск», «Стоп» и другие, формируемые автоматом управления рабочей станцией. Таким образом, *вектор выхода* в общем случае содержит как компоненты для управления внешними процессами, так и компоненты для управления внутренними процессами.
9. Введем понятие *вектора текущего состояния системы управления* – совокупность переменных состояния всех автоматов. Текущее значение вектора состояния системы управления дает полное представление о том, какие процессы выполняются в ней в данный момент времени – нет необходимости ввода дополнительных статусных переменных. Этот вектор целесообразно использовать для *отображения режимов работы оборудования* на встроенном или удаленном дисплее, а также для *протоколирования процесса оперативного управления* как в целях отладки, так и в

целях контроля за соблюдением технологии. Программа управления при использовании вектора текущего состояния системы становится *контроллепригодной*.

10. Каждый автомат может содержать любое число *вложенных автоматов*. В этом случае одна из вершин графа переходов автомата заменяется вложенным автоматом, содержащим несколько вершин. В свою очередь, любая из вершин вложенного автомата может быть заменена еще одним вложенным автоматом – рис. 3. Вложенные автоматы позволяют более наглядно представить процесс управления. Глубина вложения автоматов не ограничивается.

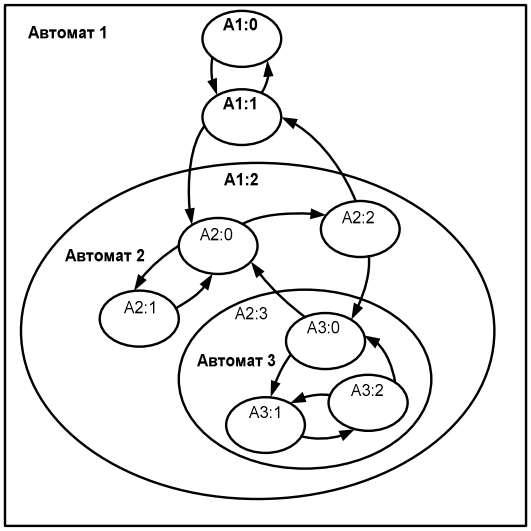

Рис. 3. Вложенные автоматы

11. *Событие* - любое изменение, которое может быть идентифицировано системой управления, способное вызвать переход автомата в другое состояние. Возможный переход отображается на графе автомата дугой и помечается *условием перехода* – логическим выражением, аргументами в котором могут быть различные события.

12. Если из текущего состояния автомата возможны несколько переходов, то все они на этапе разработки графа автомата в соответствии с техническим заданием *ранжируются по приоритету – порядку проверки условий переходов*. Приоритет проверки условия перехода задается цифрой, предшествующей условию перехода и отделенной от него двоеточием. Например, запись

    1: ПУСК*(/СТОП)

    означает, что условие перехода по нажатию кнопки «ПУСК» и не нажатию кнопки «СТОП» имеет первый, наивысший приоритет проверки.

13. При реализации автомата все возможные условия перехода из текущего состояния проверяются в порядке заданного приоритета. Если одно из условий перехода выполняется, то все остальные условия переходов, имеющие меньший приоритет, не анализируются – переход выполняется в направлении «сработавшего» условия с наивысшим приоритетом. Если ни одно из условий перехода не выполняется, то текущее состояние автомата не меняется. Такой подход позволяет минимизировать условия переходов и существенно повышает скорость обработки условий переходов в реальном времени.

14. Разделим все возможные события на *внешние* и *внутренние*. К внешним событиям отнесем следующие:

    - *Изменения состояния дискретных входов* – компонент вектора входа X, зафиксированные с помощью встроенных или удаленных плат ввода/вывода. Часть составляющих вектора входа X отнесем к *внешним управляющим*

воздействиям - командам управления преобразователем частоты и/или рабочей станцией («Пуск», «Стоп», «Реверс», «Циклический приоритет», «Основной», «Резервный»), а часть – к *состояниям внешнего оборудования* (контакторов, конечных выключателей и т.д.). Команды оперативного управления могут формироваться как кнопочными станциями, так и промышленными контроллерами и управляющими ЭВМ через встроенные модули дискретного вывода информации.

- *Сообщения*, пришедшие по каналам связи (RS-232, RS-485, CAN и т.д.) от систем управления более высокого уровня или с пультов оперативного управления (встроенных или удаленных) *с командами оперативного управления* («Пуск», «Стоп» и т.д.) или с *командами изменения конфигурации системы управления* («Местное управление», «Дистанционное управление», «Замкнутая система», «Разомкнутая система»). Будем считать, что обработка сообщений выполняется специальными обработчиками, работающими по прерываниям. Для управляющих автоматов *команды оперативного управления* и *команды изменения конфигурации системы управления* рассматриваются как обычные *компоненты вектора входа*.

К внутренним событиям отнесем:

- *Изменения состояния внутренних входов* (от встроенных датчиков), требующих незамедлительной реакции и обрабатываемых по прерываниям – *события, обрабатываемые по прерываниям*. Например, сигнал аппаратной аварии инвертора. При появлении этого сигнала сначала блокируется инвертор, а затем вызывается процедура обслуживания соответствующего прерывания. Договоримся, что в процедурах обслуживания прерываний *может производиться экстренная смена кодов текущих состояний* всех или части автоматов. Так, в приведенном выше примере, автомат управления преобразователем частоты может быть принудительно переведен в состояние «Инвертор отключен». На графах автоматов соответствующие дуги переходов будем изображать *пунктирными линиями* имея ввиду, что *переходы реализуются автоматически обработчиками соответствующих прерываний*, то есть вне контекста программы управляющего автомата.

- *Изменения значений внутренних переменных*, идентифицированные системой управления как события – *программно-обрабатываемые события*. Например: при частотном торможении двигателя текущее значение частоты стало меньше допустимого f<fmin; при регулировании давления достигнуто заданное значение давления P=Pset. Компоненты *вектора программно-обрабатываемых событий Event* будем формировать исключительно «по месту», то есть только в тех состояниях автоматов, в которых рассматриваемое событие может произойти. Так, проверка события f<fmin будет производиться только в состоянии «Частотное торможение». Это позволит отказаться от громоздкого и требующего больших затрат времени централизованного обработчика событий, запускаемого перед каждым вызовом автоматов и перейти к *распределенному контекстно-зависимому обработчику событий*. Разрешим «деятельность» по обработке событий в любом состоянии автомата. Будем считать, что эта «деятельность», если она необходима, выполняется при каждом вхождении в соответствующее состояние автомата.

- *Изменения состояния внутренних процессов*, контролируемых соответствующими автоматами. Например, подчиненный автомат перешел в заданное состояние. Для головного автомата это может означать завершение подчиненного процесса и необходимость запуска следующего процесса.

15. Если внутренний процесс имеет только два возможных состояния, то в качестве переменной состояния процесса можно использовать битовую переменную (*флаг состояния процесса*). Такая ситуация имеет место при программно-аппаратной реализации таймеров с использованием прерываний. Будем считать, что *таймеры отсчета заданных временных интервалов* запускаются внутренними выходными воздействиями - командами запуска TIMERn_GO, а величины требуемых выдержек времени задаются значениями переменных SET_Tn (число фиксированных дискрет времени, например, мс). Практическая реализация таких устройств не вызывает затруднений. Каждый раз в процедуре обслуживания прерывания от аппаратного интервального таймера проверяются работающие программные таймеры (флаги TIMERn_GO) и для всех активных таймеров декрементируется состояние переменной SET_Tn. При равенстве нулю переменной SET_Tn заданная выдержка времени считается отработанной и флаг состояния таймера TIMERn_GO очищается. Таким образом, битовые флаги TIMERn_GO могут применяться одновременно и для запуска таймеров (выходы) и для контроля за их состоянием (входы), а процесс отсчета временных интервалов может рассматриваться как *параллельный*, полностью контролируемый в процедуре обслуживания прерывания от интервального таймера: из программ дискретных управляющих автоматов достаточно лишь запускать нужные таймеры и при анализе возможных переходов контролировать флаги их текущих состояний.

16. Так как поведение управляющего автомата определяется исключительно его текущим состоянием Q[k] и не зависит от «предыстории», то реализации подлежат только те *операции*, которые должны быть выполнены *в текущем состоянии*. Кроме того, должны быть проанализированы все *условия перехода* автомата из текущего состояния в другие возможные состояния в порядке заданного приоритета проверки условий и определено *состояние перехода* Q[k+1]. Таким образом, при вызове любого автомата на одном такте, скане автомата решаются две задачи: а) «Что делать в текущем состоянии?» и б) «Нужно ли сменить текущее состояние на другое?». Это означает, что реализации подлежит только одна вершина графа переходов автомата. При этом вызов нужного фрагмента кода можно выполнить автоматически по номеру текущего состояния автомата с помощью программной конструкции, аналогичной конструкции Switch (Q[k]) языка программирования СИ:

```
Switch (Q[k])
    Case 0:
        Операции в состоянии 0;
        Определение состояния перехода автомата из состояния 0 Q[k+1];
        Выход;
    Case 1:
        Операции в состоянии 1;
        Определение состояния перехода автомата из состояния 1 Q[k+1];
        Выход;
    Case 2:
    ...
```

Эта конструкция легко реализуется на ассемблере любого процессора с помощью таблицы начальных адресов обработчиков всех возможных состояний автомата

(векторов перехода) и команды косвенной передачи управления по текущему значению вектора перехода. Подобная задача решается, в частности, в интерпретаторах команд. Пример структуры программы автомата на ассемблере сигнального процессора TMS320x240x показан ниже:

```
;Автомат управления рабочей станцией
A_W_ST:
;По коду текущего состояния автомата рассчитать адрес обработчика
                lacl    w_st            ; w_st – переменная состояния автомата
                add     #Table_vec
                tbrd    temp
                lacl    temp
;Косвенный переход на программу-обработчик текущего состояния
                bacc

;Таблица векторов перехода на обработчики состояний автомата
Table_vec:      .word   prog_st_0
                .word   prog_st_1
                …
                .word   prog_st_N

;Подпрограммы обработчиков состояний управляющего автомата
prog_st_0:
                ;Операции, выполняемые в состоянии 0
                …
                ret
prog_st_1:
                ;Операции, выполняемые в состоянии 1
                …
                ret
                …
```

17. Если число возможных состояний автомата (подавтомата) не превышает 16, то все они могут быть закодированы с помощью последовательности кодов 1, 2, 4, 8, 16, …, 32768, то есть *номером разряда переменной состояния с установленной «логической» единицей*. Такой подход к кодированию позволяет реализовать оператор переключения методом последовательного тестирования битовых переменных в слове состояния автомата и обладает еще одним преимуществом: простотой проверки условий нахождения автомата в заданных состояниях. Для этой цели можно использовать все те же команды тестирования битовых переменных, что эффективнее операций сравнения слов.
18. На *одном такте управляющей программы последовательно вызываются все автоматы, работающие параллельно*. При этом для каждого из автоматов сначала выполняются операции, предусмотренные текущим состоянием, а затем определяется состояние перехода. Если состояние перехода отличается от текущего, то оно лишь фиксируется - операции в новом состоянии выполняются на следующем такте вызова автомата.
19. Будем различать *«действия»*, которые должны выполняться однократно при изменении состояния автомата **Entry** (при каждом новом вхождении в вершину графа переходов автомата) от *«деятельностей»* **Do**, которые должны выполняться постоянно при любом вхождении в вершину графа переходов. Такое разделение целесообразно

прежде всего потому, что один раз сформированные *выходные воздействия* фиксируются во внутренних регистрах и отсутствует необходимость их постоянного обновления при неизменности текущего состояния автомата. Напротив, операции, квалифицированные как «деятельности», например, формирование в текущем состоянии автомата программно обрабатываемых событий **Event**, должны выполняться при каждом вхождении в вершину.

20. Разделим все *выходные воздействия*, которые могут формироваться в текущем состоянии автомата на три класса: *внешние, внутренние*, *сообщения:*

    - К *внешним воздействиям* отнесем изменения состояния *дискретных выходов* (компонент **вектора выхода Y**) с помощью соответствующих плат ввода/вывода, т.е реально выдаваемые во вне сигналы управления.
    - К *внутренним* – изменения состояния *битовых переменных, управляющих параллельными процессами или другими автоматами* - компонент вектора **COMMAND** (во вне не выдаются) или изменения состояний *битовых переменных*, отвечающих за *программно-идентифицируемые события* – компонент вектора **EVENT**. Частным случаем таких команд, могут быть команды разрешения или запрещения прерываний, сопровождаемые необходимыми для этой цели процедурами инициализации соответствующих периферийных устройств.
    - К *сообщениям – операции по передаче информации* (**INFORMATION**) другим устройствам по выделенным каналам связи, например, по локальной промышленной сети на базе интерфейса RS-485 или CAN-шины.

    Примеры: внешнее воздействие - сигнал замыкания «сухого контакта» реле на плате релейного ввода/вывода для включения силового контактора; внутреннее воздействие – формирование флага запуска преобразователя частоты ПУСК_ПЧ автоматом управления рабочей станцией; сообщение – активизация передачи аварийной информации на дисплей пульта управления.

21. При явном кодировании всех возможных состояний автоматов с помощью переменных состояния можно было бы вообще отказаться от битовых управляющих переменных (COMMAND), имея ввиду что автоматы могут обмениваться между собой переменными состояния. Тем не менее *разрешим формирование и опрос внутренних битовых управляющих переменных* (компонент вектора COMMAND) по следующим причинам:

    - В системах реального времени *одни и те же управляющие воздействия могут формироваться несколькими источниками*. Например, команда «Пуск» преобразователя частоты может быть подана: со встроенной клавиатуры пульта управления; с удаленного устройства верхнего уровня по одному из интерфейсов RS-232, RS-485 или CAN; от автомата управления рабочей станцией; от кнопочной станции через плату дискретного ввода и т.д. Разрешение на прием управляющего воздействия по каждому из каналов определяется текущей конфигурацией системы, в частности, возможен прием управляющего воздействия от всех источников «по ИЛИ».
    - Опрос битовой переменной при проверке возможного условия перехода производится быстрее, чем тестирование заданного состояния автомата.

22. Дискретные управляющие автоматы *могут взаимодействовать с обработчиками сообщений* по каналам связи (RS-232, RS-485, CAN), встроенными или удаленными пультами оперативного управления как *явно*, так и *неявно* – рис. 4. В первом случае автомат может инициировать передачу информации, например, при каждой смене

состояния. Во втором случае обработчик сообщений модифицирует компоненты вектора входа, в том числе внешние управляющие воздействия и биты конфигурации системы управления, а автомат при очередном вызове – обрабатывает эти воздействия.

23. *Взаимодействие* управляющих автоматов *с обработчиками прерываний неявное* и сводится к возможности принудительной смены кода текущего состояния в процедуре обслуживания соответствующего прерывания.

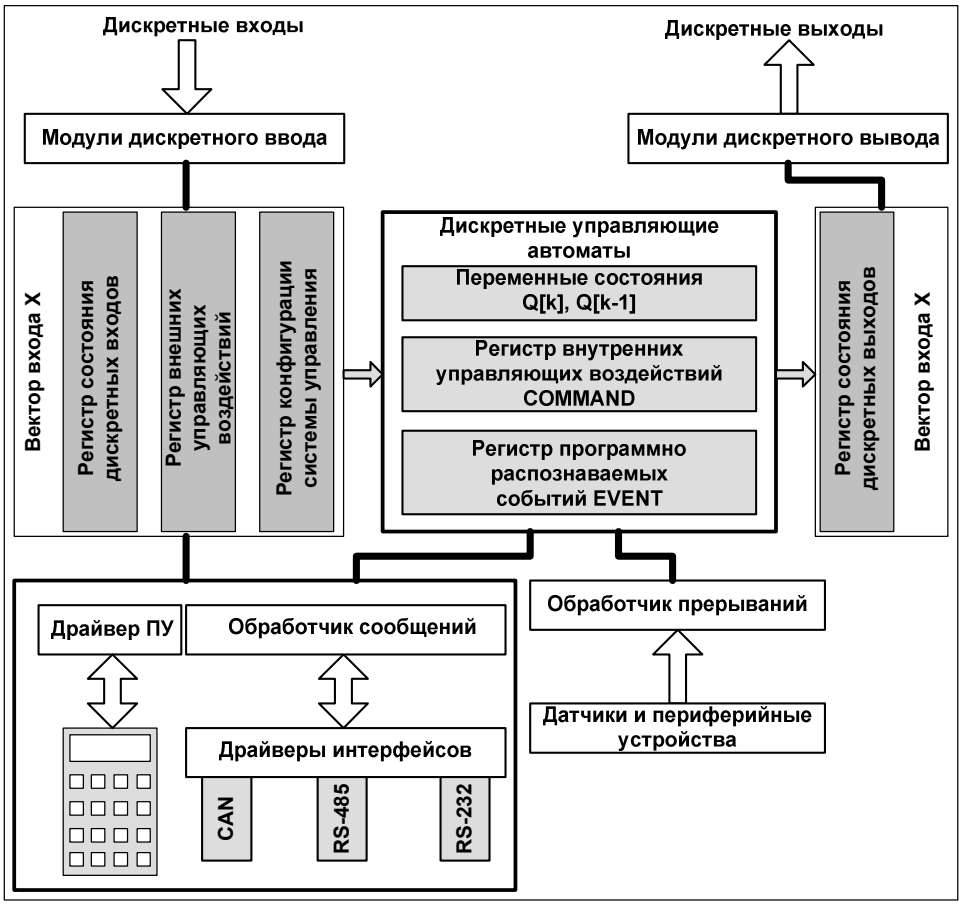

Рис. 4. Взаимодействие управляющих автоматов с обработчиками сообщений и прерываний.

24. Так как в качестве модели автомата принят смешанный автомат Мура-Мили, разрешим дополнительно выполнение операций, сопровождающих переход из одного состояния в другое – *«попутных» операций*. Такие операции будем изображать на графе переходов «в знаменателе» соответствующего условия перехода. Физический смысл попутных операций состоит в немедленной передаче информации на выход при изменении состояния входов – реализация обычной комбинационной логики, если это требуется.

25. В каждом состоянии автомата будем управлять выходами (вектором **Y**) и подчиненными процессами (вектором **COMMAND**), выполняя *«действия» исключительно при первом вхождении* **Entry**. При любом вхождении в вершину автомата будем *программно формировать возможные в текущем состоянии события* ***EVENT***, выполняя нужные для этой цели «деятельности» **Do**. Затем, будем определять *состояние перехода* Q[k+1] и, если необходимо, – реализовывать *попутные операции*, сопровождающие переход. Подпрограммы реализации *вложенных автоматов* будем вызывать при любом вхождении в вершину графа, рассматривая их как *постоянные «деятельности»*:

```
If  Q[k]≠Q[k-1]  Then  {/*Первое вхождение*/
                        Q[k-1]=Q[k];
                        Entry: /*«Действия» при первом вхождении*/
                               Управление вектором выхода Y;
                               Формирование вектора COMMAND; }
/*Последующие вхождения*/
Do:          /*«Деятельности» при каждом вхождении*/
             Формирование вектора EVENT;
             Вызов вложенного автомата, если есть;  }
/*Проверка условий переходов в соответствии с заданным приоритетом*/
If  (Условие1)  Then  {Q[k]=Q[k+1]; Попутная операция, если она есть }
If  (Условие2)  Then  {Q[k]=Q[k+1]; Попутная операция, если она есть }
…
```

26. Приведенный алгоритм обработки вершины графа переходов автомата при параллельной работе сразу нескольких автоматов обеспечивает:

    - *немедленную*, в том же такте вызова *передачу управляющего воздействия* через новое состояние автомата *всем другим автоматам, вызываемым после него*;
    - *задержанную на один такт* передачу управляющего воздействия *всем автоматам, вызываем раньше него*.

27. При составлении графа переходов автомата используется следующая нотация - рис. 5. В заголовке вершины графа указывается номер текущего состояния автомата и название состояния. В поле «Entry:» перечисляются операции, которые должны быть выполнены только при первом вхождении в вершину. В поле «Do:» перечисляются операции, которые должны быть выполнены при каждом вхождении в вершину. Рядом с дугой возможного перехода указывается логическое условие перехода. В том случае, если при выполнении перехода должна быть выполнена «попутная» операция, она указывается в «знаменателе» условия перехода. Если переходов несколько, то каждый из них может иметь свой приоритет проверки условия перехода. Если приоритет не указан, условия проверяются в произвольном порядке.

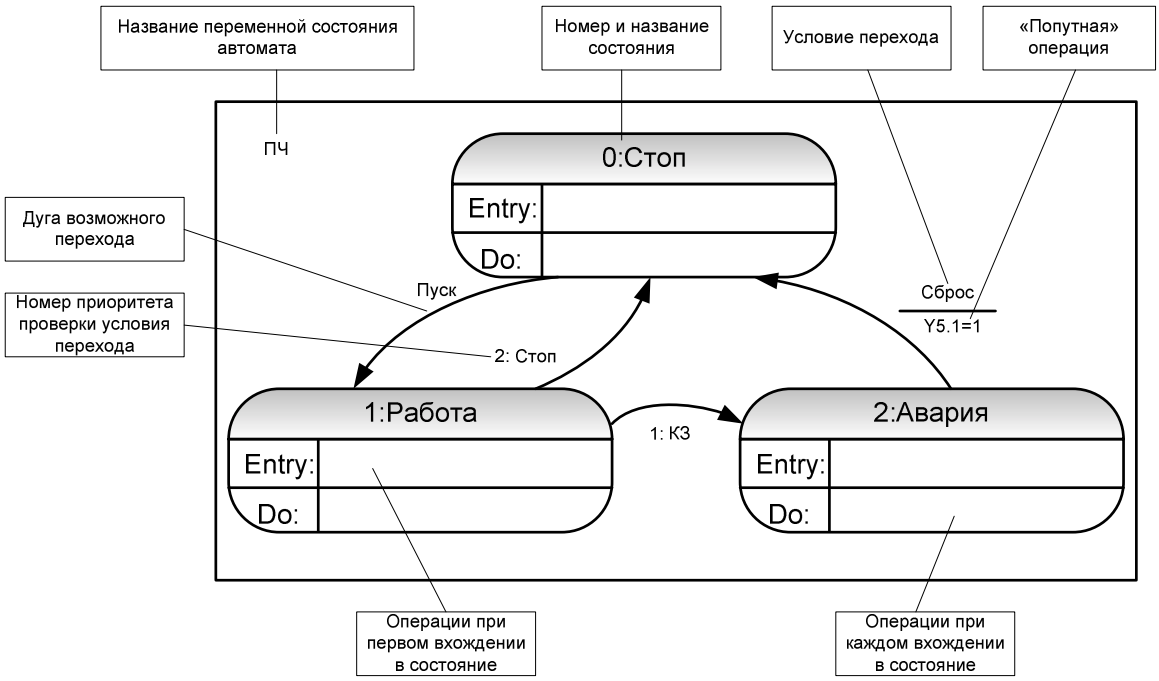

Рис. 5. Нотация, используемая в графах переходов автоматов

4. Пример использования метода

Рассмотрим автомат управления прямым пуском асинхронного двигателя в сеть. Имеются две команды оперативного управления «ПУСК» и «СТОП», которые вводятся в систему управления через плату дискретного релейного ввода (нормально разомкнутыми контактами реле). Состояние силового контактора, подключающего двигатель к сети, контролируется блок-контактом «КМ». Силовой контактор управляется контактом реле, установленным на плате дискретного релейного вывода «К» - вектор выхода Y имеет только одну компоненту «К». При возникновении аварийной ситуации выдача управляющего воздействия на силовой контактор должна быть блокирована. Выход из состояния индикации аварии допускается только по команде оператора «СБРОС_АВАРИИ». Таким образом, вектор входа X имеет четыре компоненты («ПУСК», «СТОП», «СБРОС_АВАРИИ», «КМ»). Все состояния системы управления должны быть отображены на встроенном дисплее пульта оперативного управления. Граф переходов головного автомата – управления пуском двигателя показан ниже:

Рис. 6. Граф переходов автомата управления пуском двигателя

Возможны три состояния головного автомата: два основных (двигатель отключен/подключен от/к сети) и одно аварийное, когда процесс подключения/отключения выполнить не удалось из-за неисправности контактора. Каждый раз при смене состояния автомата соответствующее информационное сообщение выводится на дисплей пульта управления. Условия переходов определяются как состоянием входов - команд оперативного управления «ПУСК», «СТОП», «СБРОС_АВАРИИ», так и состоянием подавтомата подключения двигателя к сети АПС. Подавтомат контролирует процесс включения/выключения силового контактора по состоянию его блок-контакта в течение допустимого времени срабатывания/отпускания, отсчитываемого таймером. Подавтомат вызывается как вложенный в каждой вершине графа переходов основного автомата и имеет следующий алгоритм работы:

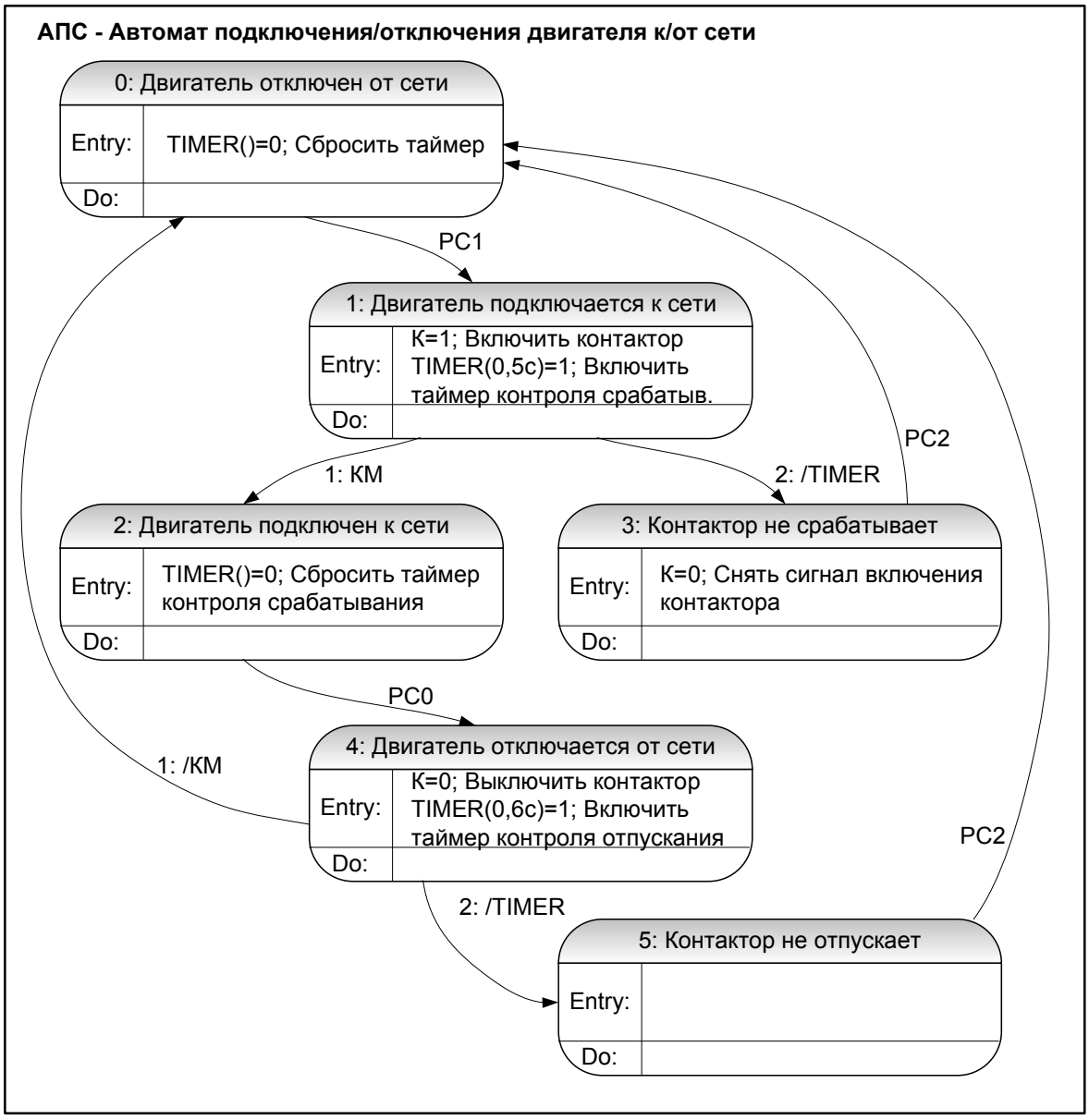

Рис.7. Граф переходов автомата подключения/отключения двигателя к/от сети

Как видно, состояния головного автомата используются в качестве входных воздействий на дугах переходов подавтомата, а состояния подавтомата – на дугах перехода головного автомата. Задание приоритета проверки условий переходов позволяет существенно упростить условия переходов. Конечно, в столь простой задаче разделение на два взаимодействующих автомата выглядит искусственно – можно было бы реализовать только один автомат. Однако, разделение становится целесообразным и необходимым, если предположить, что подчиненный автомат физически работает на удаленной интеллектуальной плате ввода/вывода, соединенной с основной системой управления только каналом связи, например, CAN-шиной.

В этом случае при каждом вхождении в вершину головной автомат должен дополнительно инициировать передачу сообщения по каналу связи подчиненному автомату о изменении своего состояния. Аналогично, подчиненный автомат при каждой смене своего состояния должен уведомить об этом головной автомат, послав соответствующее сообщение по шине.

Если команды управления «ПУСК», «СТОП», «СБРОС-АВАРИИ» вводятся через плату ввода/вывода основной системы управления, поступают с пульта оперативного

управления или по одному из интерфейсов, то передавать состояние четвертого входа «КМ» из удаленной системы управления нет необходимости. Подавтомат будет не только сам контролировать вход «КМ», но и выдавать управляющее воздействие «К» на силовой контактор. Тем самым полностью реализуется принцип распределенного микропроцессорного управления. Если же указанные выше команды управления вводятся через удаленную систему, то любое изменение состояния этих входов должно сопровождаться передачей соответствующего сообщения основной системе управления, что технически не вызывает трудностей.

5. Заключение

Рассмотренный метод реализован в отечественной серии преобразователей частоты «Универсал» как для управления режимами работы преобразователя, так и для станций группового управления насосами, в том числе на основе интеллектуальных плат ввода-вывода с сетевым CAN-интерфейсом. Главное преимущество метода – полное документирование алгоритма на основе функциональной декомпозиции задачи управления, простота реализации и сопровождения программного продукта. Современные процессоры для встроенных применений имеют как специальные команды для эффективной реализации операторов переключения, так и средства работы с битовыми переменными для проверки условий переходов, что обеспечивает предельную компактность кода и высокое быстродействие программ реализации дискретных управляющих автоматов.

Литература.

1. Козаченко В.Ф., Грибачев С.А. Перспективная серия микроконтроллеров фирмы Texas Instruments ‘240x для систем цифрового управления двигателями.
2. Козаченко В.Ф., Темирев А.П., Обухов Н.А., Анучин А.С. и др. Контроллеры МК11.3 для высокопроизводительных систем прямого цифрового управления двигателями. CHIP NEWS. Инженерная микроэлектроника, №4(67), апрель 2002г. с. 24-30
3. А.А. Шалыто. Алгоритмизация и программирование для систем логического управления и «реактивных» систем. «Автоматика и телемеханика», 2001г., №1, с.3-39.
4. А.А. Шалыто, Н.И. Туккель. Switch-технология – автоматный подход к созданию программного обеспечения «реактивных» систем. «Программирование», 2001г., №5, с.45-62
5. Козаченко В.Ф., Обухов Н.А., С.А. Трофимов, П.В. Чуев. Применение DSP-микроконтроллеров фирмы Texas Instruments в преобразователях частоты «Универсал» с системой векторного управления. Электронные компоненты. №4 2002г., с. 61-64.