Принципы программирования контроллеров на языке функциональныхблоковыхдиаграмм

Б.Г.Севастьянов,И.А.Жолобов,Д.Б.Севастьянов

Для повышения эффективности разработки и эксплуатации автоматизированной системы управления технологическим процессом (АСУ ТП) программисты, реализующие программы на языке функциональных блоковых диаграмм или алгоблоков (FBD), должны придерживаться определённых принципов. На необходимость выполнять работы с учётом принципов построения АСУ ТП указывал в своё время и академик Глушков В.М. [1]. Обращается внимание на принципы построения систем и в других работах [2].

Ниже излагаются принципы программирования промышленных контроллеров на языке FBD и одном из вариантов его реализации языка – графическом языке непрерывных функциональных схем (CFC), знание и применение которых уменьшает ошибки при программировании, упрощает отладку программ, значительно снижает трудоёмкость при эксплуатации системы. Языки программирования описываются в различных источниках, например [3 – 7]. Язык функциональных алгоблоков FBD и язык непрерывных функциональных схем CFC рекомендованы Международной Электротехнической Комиссией в стандарте IEC 61131.3 [8].

Принципы уменьшают трудоёмкость разработки программ, состоящих из десятков или сотен алгоблоков. Если программа составляется на языке функциональных алгоблоков, то целесообразно придерживаться следующих принципов:

1. Принцип функциональной полноты. Алгоритм и программа контроля и регулирования должны максимально реализовывать функции объекта управления (ОУ). Сюда входит учёт введения новых функций или задач. Введение очередной функции не должно изменять (существенно)

структуру программы. Структура программы должна содержать проверку входной и выходной информации на достоверность, отражать логику нормального функционирования ОУ, предусматривать анализ предаварийных состояний и аварийных ситуаций, содержать блоки прогноза вероятных нарушений и формирования рекомендаций обслуживающему персоналу или операторам технологического процесса, передачу и приём информации по локальной сети, связь с верхним уровнем (с автоматизированным рабочим местом, например, оператора-технолога).

2. Принцип единообразия в шифрации, нумерации и расположении алгоритмических блоков (алгоблоков), выполняющих одну или несколько взаимосвязанных функций. Расположение и их нумерация должны отражать последовательность преобразования информации. По-другому этот принцип можно назвать принцип идентичности нумерации блоков в группах. Нумерация каждой группы блоков, реализующих одинаковые функции, должна быть идентична. Например, первая группа имеет нумерацию от 10 до 19, тогда вторая – от 20 до 29 и т.д. На это обращается внимание в работах [3, 9], когда управление осуществляется десятками задвижек.

3. Принцип нежёсткой нумерации. Между участками программы предусматривают фиктивные блоки (не более трёх-пяти), которые позволяют в будущем вносить изменения фрагмента программы без изменения нумерации алгоблоков во всей программе. Такой приём назовем принципом нежесткой нумерации.

4. Принцип вероятного изменения модификатора. В большинстве случаев модификатор определяет количество алгоритмов в одном алгоблоке. При расположении алгоблоков друг под другом следует учитывать возможность изменения размера модификатора в сторону увеличения (на один размер). Этот принцип работает в тех средах и на тех контроллерах, где введён модификатор и его суть совпадает с сутью, которая заложена в понятие модификатор в контроллерах Р-130, КР-300, КР-500 [4, 6].

5. Принцип красоты геометрического расположения алгоблоков. Чтобы уменьшить число линий, связывающих алгоблоки, (не превращать программу в паутину) можно несколько параллельных линий объединять в одну, формируя информационный кабель. *Информационный кабель* в программе внешне неотличим от других линий, но при подводе курсора должно появляться падающее меню. В меню указываются все связи, которые проходят по этой линии. Границы группы алгоблоков располагают на одной линии, чтобы можно было проводить информационные кабели без дополнительных зигзагов и пересечений. Повторяющиеся фрагменты программы должны иметь аналогичное расположение алгоблоков.

6. Принцип промежуточного программного клеммника. Введение в программу промежуточных алгоблоков при вводе информации, или по-другому их назовём промежуточные «клеммники». Данный приём программирования позволяет оперативно переходить на резервный канал, не корректируя связи в самой программе.

Поясним несколько принципов на примерах. Возьмём контроллер Ремиконт Р-130 [3, 4], что не принципиально. Программы в контроллере Р-130 пишутся на языке функциональных алгоблоков. Вся оперативная память контроллера условно (виртуально) разбивается на зоны памяти, которые называют алгоблоками. Структуру алгоблока, приведённую на рис. 1, можно считать классической. В контроллере Р-130 имеется в ОЗУ 99 зон памяти (виртуальная память), которые названы алгоблоками. В каждый алгоблок из постоянного запоминающего устройства (ПЗУ) может вызываться любой алгоритм. Набор алгоритмов называют библиотекой алгоритмов. Каждому алгоритму присвоен свой код (цифра) и шифр (буквенное обозначение). Например, алгоритм И, реализующий логическую операцию И, имеет код 70, а шифр — И (в контроллере Ремиконт Р-130 двухвходовая логическая операция И названа ЛОИ). Связи между алгоблоками осуществляют адресно или графически, т.е. линиями.

На рис. 1 изображён алгоблок с алгоритмом И. В одном алгоблоке находятся два алгоритма И. Шифр алгоритма И – И, код этого алгоритма равен 70. Это двухвходовая логика, т.е. два входа и один выход.

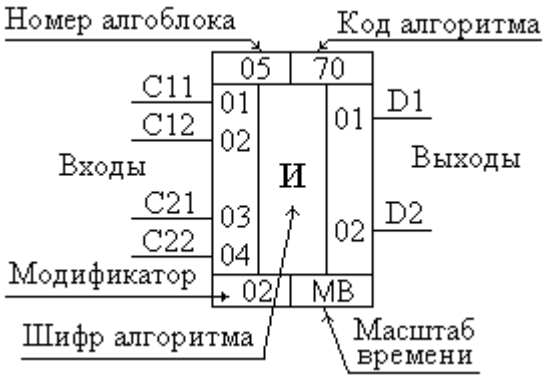

Рис. 1. – Структура алгоблока

Для первого алгоритма И входы обозначаются C11 и C21. Выход – D1. В данном случае модификатор равен двум и указывает, что в алгоблоке номер пять размещено два алгоритма И. В одном алгоблоке, согласно документации [4], может быть до 20-ти алгоритмов И.

Приведём условную последовательность алгоблоков, например, реализующих управление задвижкой [9]. Рассмотрим принцип единообразия в шифрации, нумерации и расположении алгоритмических блоков на примере управления двумя задвижками Z1 и Z2 (рис. 2). В каждый алгоблок A1—A15, A21—A35 помещён из ПЗУ определённый алгоритм. Допустим в алгоблоке A1 находится элемент И, с помощью которого осуществляют защиту от ввода некорректной команды. Например, блокируется команда открыть, когда задвижка открыта. В алгоблоке A2 находится таймер, контролирующий время открытия задвижки. При такой нумерации алгоблоков и алгоритмов достаточно описать программу для одной задвижки. Если взять теперь вторую задвижку, то несмотря на то, что с первого взгляда они имеют другую нумерацию суть та же самая и имеем аналогичную нумерацию алгоблоков и алгоритмов. Возьмём алгоблок A22. Если отнять 20, то получим A2. Это тот же таймер, контролирующий открытие задвижки, только не первой задвижки Z1, а второй Z2.

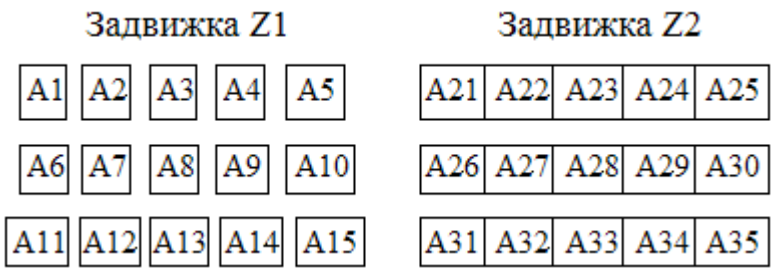

Рис. 2. – Иллюстрация применения второго и третьего принципа.

Для второй задвижки алгоблоки на рисунке не разъединены. Это сделано для компактности, что не искажает суть.

Рассмотрим следующий принцип: принцип промежуточного программного клеммника. Приведём структуру программы в общем виде (рис. 3). Допустим, *в нашем случае резервным каналом является восьмой канал (X8)*. **Вкачествепромежуточногоклеммникадлякаждогодискретного сигнала используют алгоритм ИЛИ, а для аналоговых сигналов может использоваться алгоритм масштабирования с коэффициентом масштаба, равномединице.**

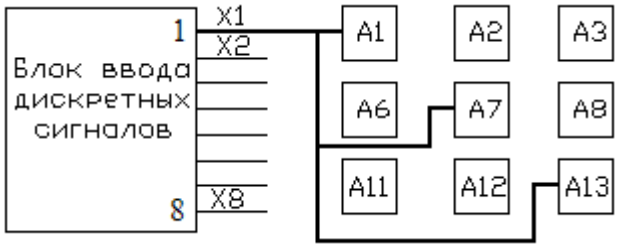

Рис. 3. – Фрагмент программы

Сигнал первого канала поступает (конфигурируется) на несколько алгоблоков: A1, A7 и A13. Если первый канал откажет, то сигнал первого канала переводится на резервный восьмой канал. В этом случае необходимо будет изменить конфигурацию в трёх алгоблоках. А если программа более сложная, то число изменений может быть больше. Если система работает в режиме реального времени и объект повышенной опасности, то изменения в программе требуется делать быстро и точно. Для этого требуется хорошее знание алгоритма работы программы, качественную сопроводительную документацию, чтобы быстро внести изменения. Как показал опыт, на практике это не всегда происходит корректно. При такой спешке

программист может не восстановить одну из связей. Отсутствие любой связи приводит к сбою или отказу в работе программы. Отказ программы приводит к отказу определённой системы в АСУ ТП. Для исключения внесения ошибок при работе в экстремальной ситуации предлагается в программе использовать промежуточный программный клеммник.

На рисунке 4 представлена та же программа, только с промежуточным программным клеммником.

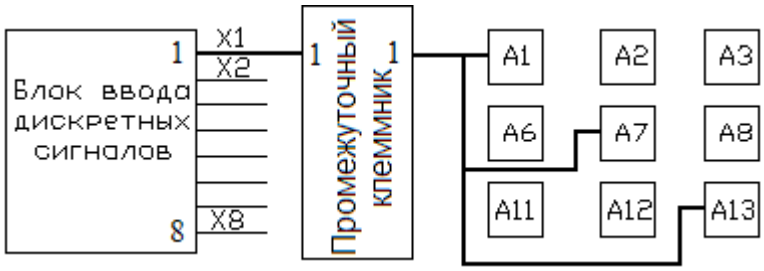

Рис. 4. – Фрагмент программы до отказа первого канала

В случае отказа первого канала переходят на резервный восьмой канал аппаратно и программно. На рис. 5 показаны изменения в программе при наличии клеммника при переходе на резервный канал.

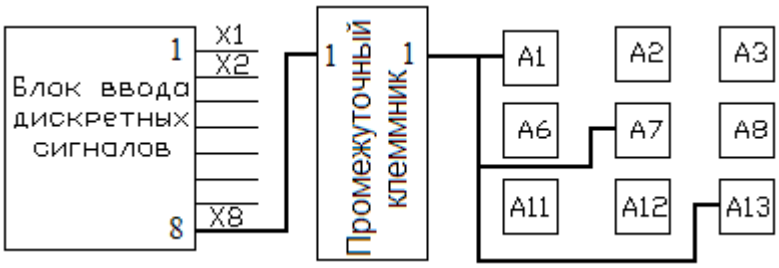

Рис. 5. – Иллюстрация перехода на резервный канал

При наличии промежуточного клеммника делается, как видим одно изменение (одна конфигурация), не меняя ничего в самой программе. Такое изменение программы не требует высокой квалификации от программиста.

Чтобы был понятен четвёртый принцип, поясним понятие модификатор на примере двухвходовой логики И (рис. 1). Допустим, в одном алгоблоке может находиться до 20 элементов И, но в нашей задаче требуется всего два элемента И. В процессе работы может возникнуть необходимость увеличить количество алгоритмов И именно в этом алгоблоке, например, ещё на два (этот принцип перекликается с принципом непрерывного развития

системы [1]). Но увеличение количества алгоритмов в одном алгоблоке увеличивает его размер. Если алгоблоки расположены достаточно плотно, то это вызовет необходимость перекомпоновки части схемы. Надо ли везде предусматривать увеличение размера алгоблока или вставки в это место дополнительного алгоблока. Конечно, нет. Это следует делать в тех местах, где с большой вероятностью появится такая необходимость, где у разработчика имеются сомнения в окончательности принятого решения.

Рассмотрим принцип красоты геометрического расположения алгоблоков. В больших проектах имеются много контуров регулирования с использованием, например, ПИ-регулятора, одинакового алгоритма управления десятками задвижек (программы, в основном, имеют одинаковую структуру и отличаются только настройками) и т.д. Перед началом программирования следует определиться с общей структурой регулятора [10], с общей структурой управления задвижкой [9]. И тогда, глядя на большую программу на языке FBD или CFC, становится понятно — это регулятор, а это — блок управления задвижкой.

Если имеется резервное технологическое оборудование, например, два насоса: основной и резервный, то нельзя параметры основного и резервного оборудования вводить в один контроллер, через который осуществляется контроль и управление. При отказе этого контроллера происходит отказ и системы управления оборудованием, несмотря на то, что имеется резервное оборудование. В этом случае должно быть предусмотрено резервирование и аппаратных средств (контроллеров, блоков питания) или параметры резервного оборудования должны быть заведены на другой контроллер. В случае «горячего» резерва контроллеров в них должна функционировать и соответствующая операционная система.

Если управление производится объектом с взаимосвязанными параметрами, то в этом случае группа взаимосвязанных параметров должна подключаться к одному контроллеру. В этом случае отказ любого контроллера, работающего в локальной сети, не будет влиять на

работоспособность другого контроллера (другой программы), или это влияние будет не существенным. Такой подход можно назвать принципом максимальной информационной автономности реализуемой задачи или *принципом минимальной независимости* одного контроллера от другого.

Вполне возможно, что многие эти принципы разработчики используют подсознательно. Надеемся, эти принципы помогут специалистам, разрабатывающим и эксплуатирующим программное обеспечение. Такой подход нашёл поддержку и одобрение у специалистов автоматчиков, обслуживающих АСУ ТП, где приходилось внедрять различные системы. Это видно в статьях [11, 12].

Систематическое применение принципов при программировании дисциплинирует разработчика, повышает качество программ. Со временем у программиста вырабатывается свой «подчерк» написания программ.

Программы, разработанные с учётом изложенных принципов, повышают надёжность и качество автоматизированных систем контроля и управления.

Списоклитературы:

1. Глушков В.М. Введение в АСУ. -Киев: Техніка, 1972.–310с.
2. Васильев С.Н. От классических задач регулирования к интеллектуальному управлению // Изв. РАН. ТиСУ. 2001. №2.–с.5–21.
3. Севастьянов Б.Г. Реализация дискретных систем управления на контроллерах. -Учебное пособие. Гриф УМО. – Волгоград, 2012. -230с.
4. Певзнер В.В., Лахова Н.В., Никольская И.В., Прохорова Н.И. Микропроцессорный контроллер Ремиконт Р-130. - М.: Типография НИИтеплоприбора, 1990. – 330с.
5. Berger Hans, Automating with STEP 7 in LAD and FBD. – 2012. – 451 p.

6. Плескач Н.В., Макаров С.К., Макаров В.Н. Промышленные контроллеры для распределенных систем серии Контраст// Промышленные АСУ и контроллеры. -1999.- №2.-С.48-52.

7. Петров И.В., "Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования" / Под ред. проф. В.П. Дьяконова. -М.: СОЛОН-Пресс, 2004. - 256с.

8. Karl-Heinz John, Michael Tiegelkamp., IEC 61131-3: Programming Industrial Automation Systems. – Springer, 2001. – 376 p.

9. Севастьянов Б.Г. Микропроцессорное управление задвижками, распределяющими потоки жидкости и газа // Приборы и системы. Управление, контроль, диагностика. -2008.- №10.-С.1-5.

10. Севастьянов Б.Г. Безударность и надёжность систем автоматического регулирования// Приборы и системы. Управление, контроль, диагностика. -2007, №12.-с.1-4.

11. Севастьянов Б.Г., Жолобов И.А. Алгоритм таймера пользовательской библиотеки [Электронный ресурс] // «Инженерный вестник Дона», 2013. №4. – Режим доступа: http://www.ivdon.ru/magazine/archive/n4y2013/2207 (доступ свободный) – Загл. с экрана. – Яз. Рус.

12. Аль-Хулайди А.А., Чернышев Ю.О. Разработка параллельного алгоритма нахождения оптимального решения транспортной задачи на кластере [Электронный ресурс] // «Инженерный вестник Дона», 2011. №2. – Режим доступа: http://www.ivdon.ru/magazine/archive/n2y2011/445 (доступ свободный) – Загл. с экрана. – Яз. Рус.