lws0954 28 августа в 22:48

Вот, как просто! Автоматы в деле. Для ПЛК фирмы DELTA

Параллельное программирование*, Промышленное программирование*

Промышленный логический контроллер (ПЛК) - это тот же компьютер, но попроще. В нем есть все или почти все, что есть в любом ПК, но только, может, в меньшем объеме или не такой производительности. Но зато он может работать там, где обычный компьютер неприменим. У ПЛК есть то, что делает работу с ним проще при управлении оборудованием. Например, наличие "на борту" каналов ввода/вывода дискретных логических сигналов. Его программирование специфично. Выбор языков программирования достаточно ограничен, по большому счету их всего-то пять, и определяется стандартом МЭК 61131-3 [1]. И этого, как убеждает практика, по большому счету вполне достаточно.

Выбор ПЛК фирмы DELTA, кроме наличия собственной IDE, предоставляет доступ к широкому перечню периферийного оборудования. Фирменное ПО, как минимум, удобнее тем, что не требует особой настройки и «в один клик» работает на всей линейке технических средств фирмы. Минусы могут проявиться в отставании от передовых тенденций программирования. Но для ПЛК это не самая большая проблема, т.к. языки, определяемые стандартом, достаточно консервативны, а их настройка под разные типы ПЛК, как правило, не так уж сложна.

Можно даже утверждать, что тип ПЛК достаточно условен, т.к. программирование при наличии промышленного стандарта для них фактически неотличимо (различие в IDE пока не рассматриваем). По крайней мере, сам стандарт на это настраивает. Нам же далее будет важнее реализация определенной идеи. И если уж, как мы увидим, с этим справится столь элементарный язык программирования, как язык релейно-контактных диаграмм, то это будет вполне по силам и любому другому языку программирования для ПЛК. И уж тем более по плечу почти любому из известных языков для ПК.

Краткое введение в IDE

Установка IDE фирмы DELTA ISPSoft не требует от программиста каких-либо усилий. Установив ее, выберем из всего множества [стандартных] языков язык релейно-контактных схем (LD) и функциональных блоков (FB). На рис. 1 приведен пример проекта в среде ISPSoft.

В рамках программного проекта IDE код ПЛК разбивается на программные модули – POU (Program Organization Units), которые могут быть трех типов.

1. Программа – PROG (в IDE модуль имеет пометку PRG).
2. Функциональный блок - FB.
3. Функция - FC.

Дерево проекта содержит типовые узлы, среди которых Программы и Функциональные блоки для нас представляют наибольший интерес. Узел Программы содержит исполняемые модули типа PRG (тип программного модуля и язык, на котором он реализован, указаны правее имени модуля в дереве проекта). Программные модули узла Программы содержат исполняемый программный код и запускаются последовательно сверху вниз. Есть несколько режимов работы, на которые можно настроить их работу. Это циклическое исполнение модуля, запуск его по времени, и по прерыванию.

Каждый из программных модулей разбивается на Цепи или Командные строки (Network), где каждая строка представляют код на языке LD. Он включает и программный вызов функциональных блоков на языке FB. Цепи исполняются сверху вниз, а код в пределах цепи – слева направо. И эта та специфика, которая кардинально отличает работу «программной схемы» от аналогичной по виду электронной схемы, элементы которой работают параллельно. И наша задача эту специфику, если не устранить, то хотя бы в чем-то смягчить.

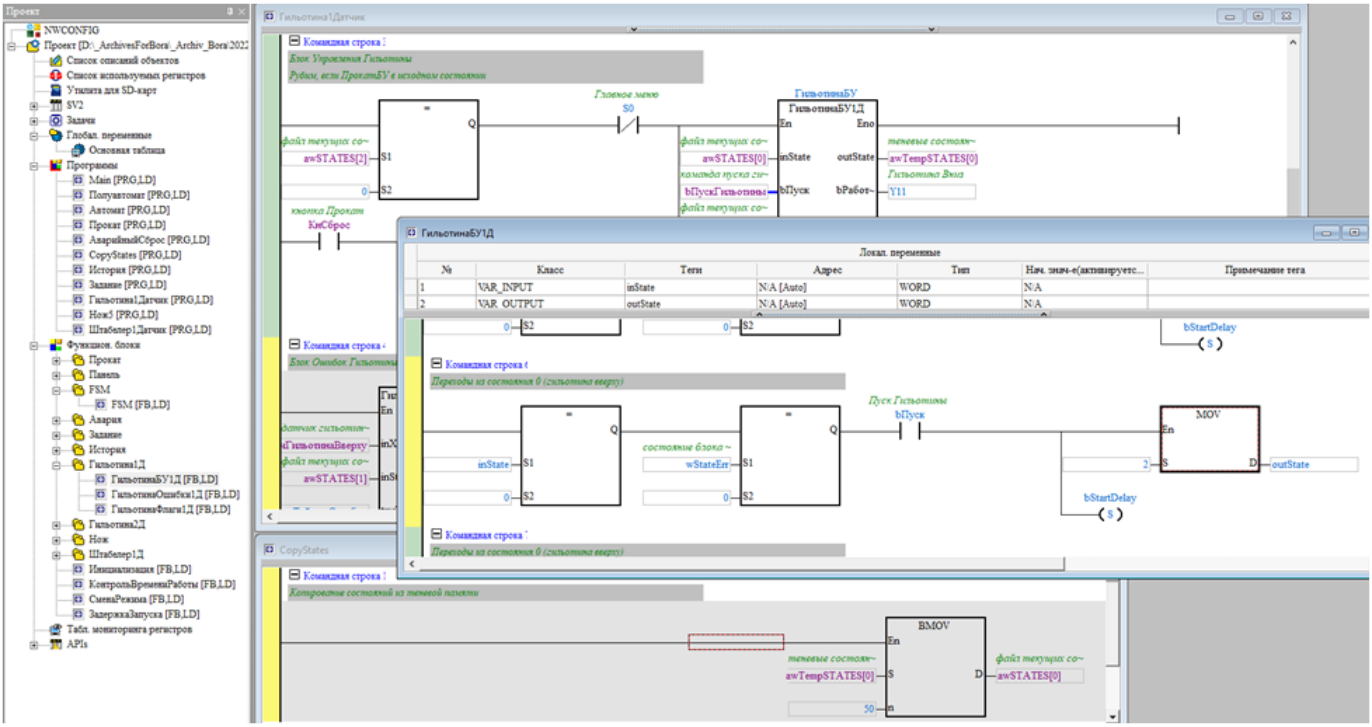


Рис.1. Пример проекта в ISP Soft

Описанная выше структурная организация программ ПЛК, как и сам язык программирования LD, будут непривычны для программиста, знакомого с обычными языками программирования. Хотя на самом деле принципиальной разницы между ними нет. Особенно с учетом описанной специфики работы программных модулей и кода внутри них. Ведь, и там и там речь идет фактически об обычном последовательном

программировании лишь с разной формой, в общем-то, типичных операторов. А если

учесть, что язык LD включает операторы типа пересылки данных, сравнения, вызова функциональных блоков и т.п., то такая связь становится еще очевиднее.

Однако, если окунуться в историю создания и развития ПЛК, то их языки программирования изначально были рассчитаны на технических специалистов, далеких от программирования. Другими словами, специалист, который знает, что такое реле, должен был уметь программировать, используя понятные ему вещи. И здесь таится тот когнитивный диссонанс, преследующий программиста под ПЛК слабо разбирающегося в программировании, но прекрасно понимающего работу электронных схем. Поэтому надо понимать то удивление или непонимание, которое может у него возникнуть при сравнении результатов работы программы для ПЛК и аналогичной ей электронной схемы (предположим, что в идеале и то и другое состоит из одних только реле).

Причина отмеченного выше противоречия достаточно очевидна: компоненты программы ПЛК работают последовательно, а компоненты электронной схемы – параллельно. Можно ли устранить эту разницу и, если можно, то как? Ответ следующий: пусть не полностью, но можно. И поможет нам в реализации этого автоматная модель программ и технология автоматного программирования (теоретическое обоснование его представлено в предыдущей моей статье [3]).

Теория – это, конечно, правильно и хорошо, но вопрос, как ее реализовать, используя достаточно скромные возможности [последовательных] языков программирования для ПЛК? И поскольку «лучше раз увидеть, чем сто раз услышать» рассмотрим для этого программную реализацию RS-триггера на ПЛК. Его модель, как и автоматные модели отдельного элемента И-НЕ и всего триггера, приведены в упомянутой выше статье.

Реализация RS-триггера

Пример проекта реализации модели RS-триггера приведен на рис. 2. Он содержит три программных модуля. Это два модуля типа PRG – RS_триггер и CopyStates и один типа FB – И-НЕ.

В модуле RS_триггер первые две цепи просто инвертируют два установочных входных сигнала триггера. Инвертирование необходимо, чтобы в исходном состоянии они были в единичном состоянии (кнопки, имитирующие соответствующие сигналы, отжаты). Так триггер фиксируется в неком установившемся состоянии и при этом не нужно будет держать кнопки, имитирующие сигналы, в нажатом состоянии. Переменные bX1 и bX2 будут внутренними локальными переменными, зависящими от состояния внешних входных сигналов ПЛК, соответственно сигналов от кнопок X1 и X2.

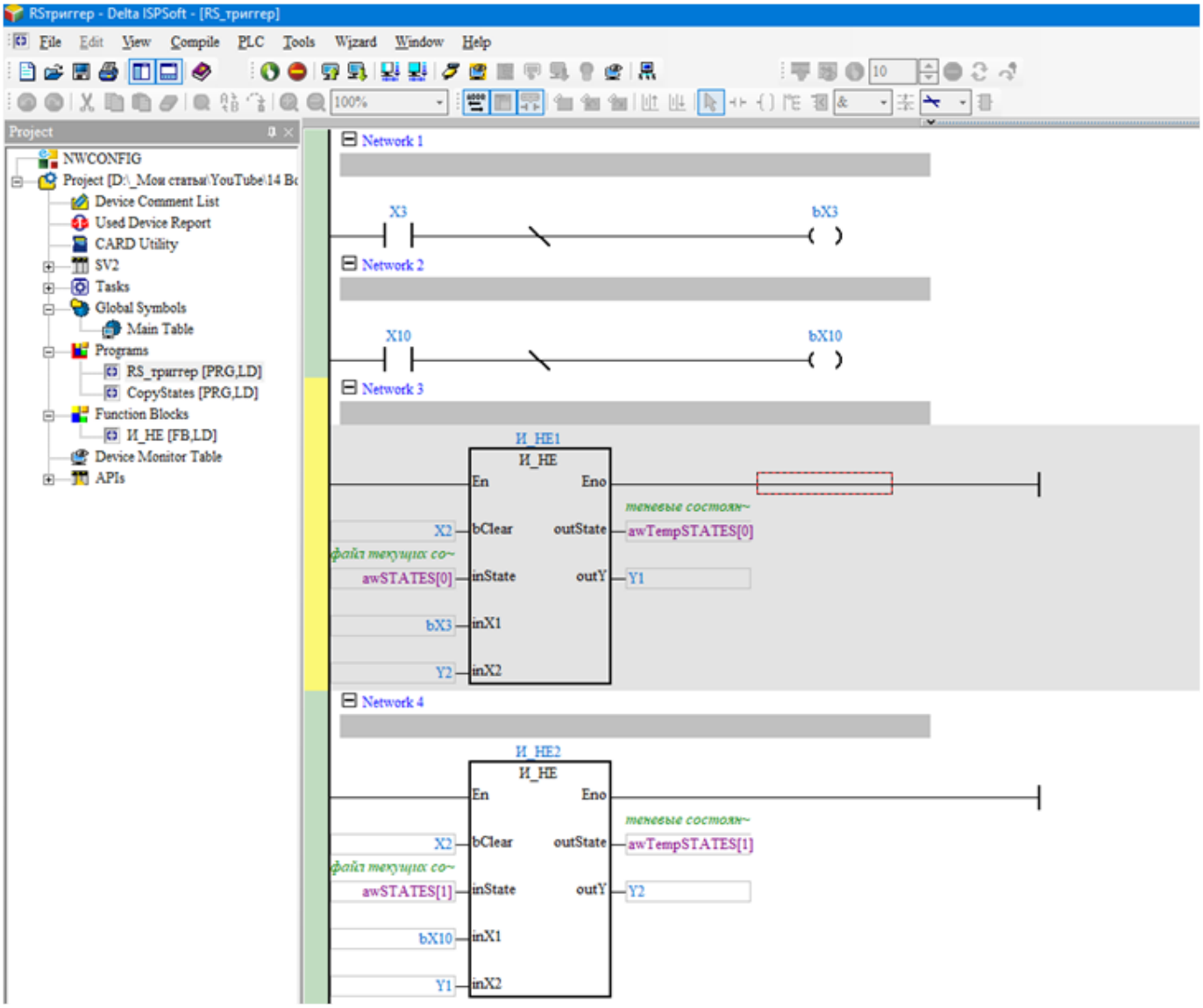


Рис. 2. Проект RS-триггера в ISP Soft

На рис.3. приведена реализация программного модуля CopyStates. Его функция – на каждом цикле работы ПЛК копировать массив слов с именем awStates в массив с именем awStates. Эти массивы содержат сопоставленные друг другу теневые и текущие состояния программных автоматов. Каждому автомату соответствует один элемент массива текущих состояний и сопоставленный ему (с таким же номером) элемент массива теневых состояний. Каждый автомат в процессе функционирования воспринимает элемент массива текущих состояний (это и есть его текущее состояние), и устанавливает в ситуации перехода значение следующего состояния, помещая его значение в элемент массива теневых состояний. Вот собственно и весь механизм реализации функции переходов отдельного автомата и множества параллельных переходов автоматов некой автоматной сети.

Если бы в рамках той же SWITH-технологии аналогичным образом устанавливались текущие состояния автоматов, то это позволило бы рассматривать и в ней параллельную работу автоматов на уровне состояний автоматов. Хотя для корректной реализации параллелизма нужно помещать в теневую память и данные. Однако в нашем случае мы

параллелизма нужно помещать в теневую память и данные. Однако в нашем случае мы этот механизм оставляем для реализации самому программисту. Даже в силу того, что только он знает, какие данные будут использованы для синхронизации автоматных процессов и, исходя уже из этих соображений, выбирать механизм оперирования ими (один из них мы далее рассмотрим).

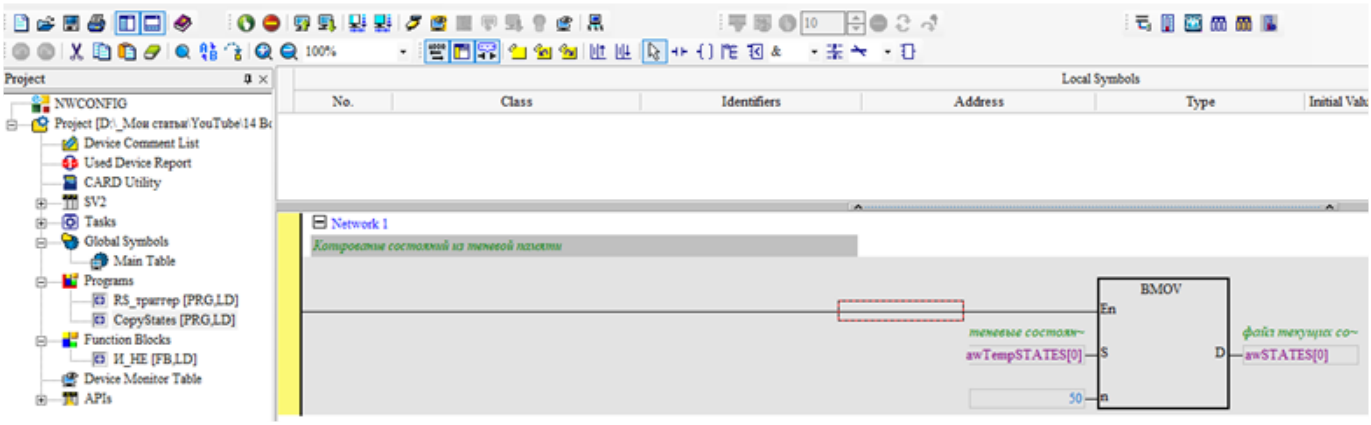


Рис. 3. Программный модуль CopyStates

Итак, реализация автоматов сводится к следующим шагам. Во-первых, создаются массивы для текущих и теневых состояний. Элементы этих массивов назначаются отдельным автоматам: один элемент для отражения текущего состояния и элемент с таким же номером из массива теневых состояний. Программный автомат, находясь в некотором текущем состоянии, анализирует входные сигналы, глобальные и локальные переменные, текущие состояния других автоматов и т.д. и т.п. В ситуации перехода он выполняет назначенные этому переходу действия и устанавливает, если это необходимо, новое текущее состояние, используя для этого массив теневых состояний.

Описанный процесс реализации функций переходов и выходов происходит последовательно во всех автоматах проекта. И только после этого массив теневых состояний копируется в массив текущих состояний. Это и реализует как раз модуль CopyStates. Далее этот цикл повторяется.

Реализация модели элемента И-НЕ

Модель элемента И-НЕ реализована в форме функционального блока (FB). Его программная реализация на языке LD приведена на рис. 4. Переменные, назначенные функциональному блоку, представлены на рис. 5 (на рис. 4 они скрыты).

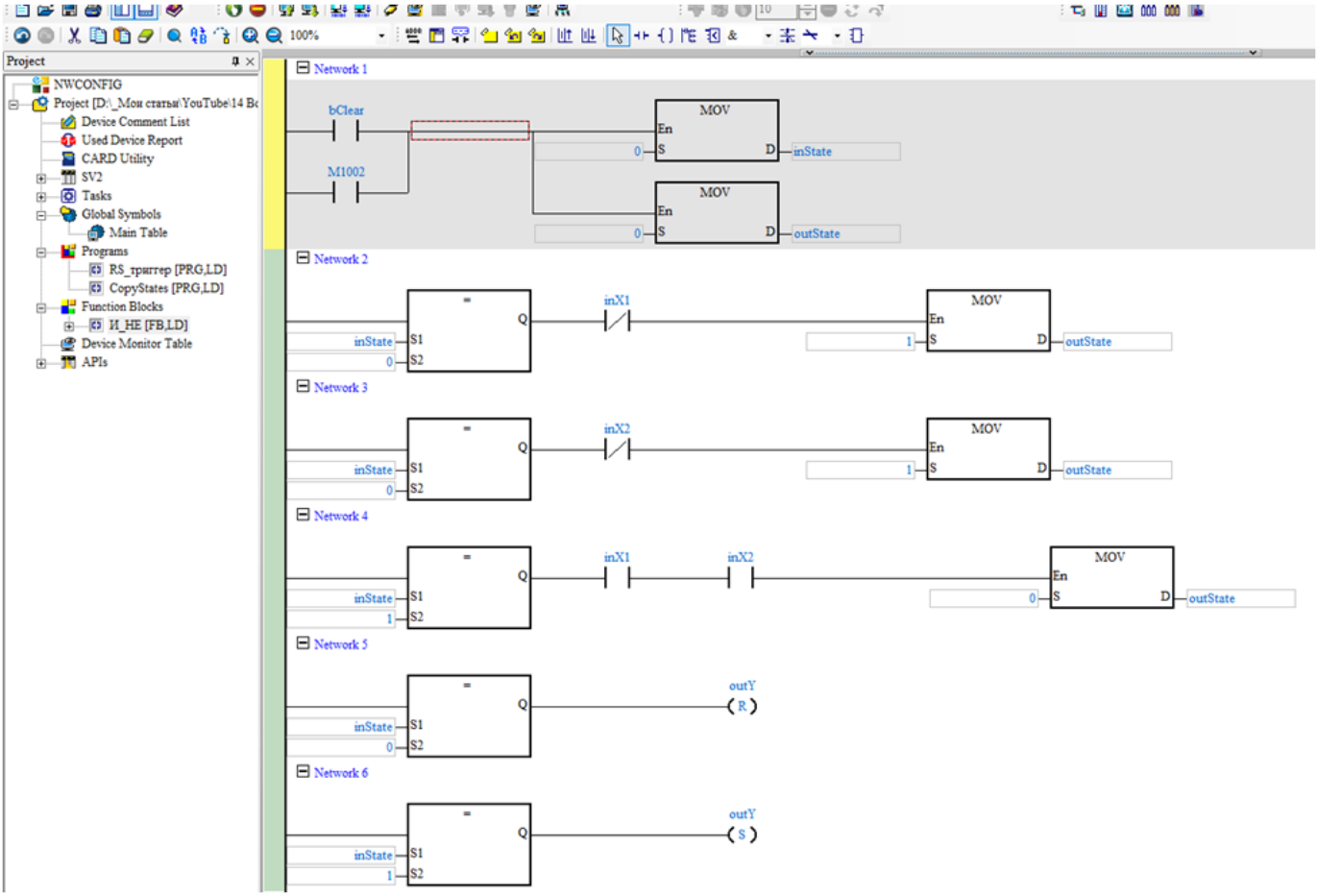


Рис. 4. Реализация элемента И-НЕ на языке LD

Любой FB может иметь переменные нескольких типов: локальные переменные – тип VAR, входные переменные - VAR_INPUT, выходные – VAR_OUTPUT и их совмещенный тип – VAR_IN_OUT. Наша программная реализация элемента И-НЕ имеет только входные и выходные переменные. На диаграмме они представлены входными/выходными линиями соответствующих FB (см. FB с именами И_НЕ1, И_НЕ2 рис. 2). Таким образом, функциональные блоки представляют собой некий аналог, близкий понятию подпрограммы с параметрами в форме его входных и выходных каналов.

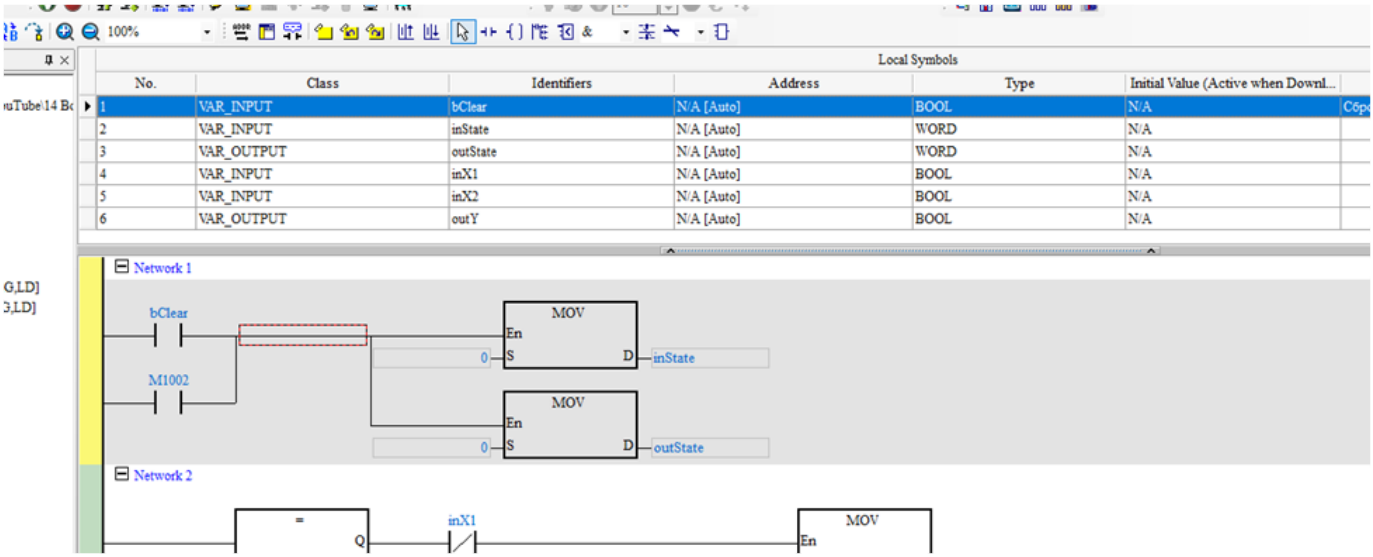


Рис. 5. Переменные FB И_НЕ

Цепь 1 (Network 1) в реализации модели элемента выполняет стандартную операцию для любого функционального блока в автоматной форме. Она устанавливает автомат в исходное состояние по сигналу сброса – bClear или при установке системной переменной M1002, принимающей однократно на один цикл работы ПЛК единичное значение при включении ПЛК.

Тестирование и эксперименты

После записи программы в ПЛК она начинает свою работу, сразу входя в режим генерации. Это процесс отражают светодиоды выходов Y1, Y2. Нажатие кнопок установочных входов, приводит модель в то или иное устойчивое состояние. Ввести модель в режим генерации можно, нажав кнопку сброса – X2 (см. рис.2). И это то, что ожидалось от поведения данной модели в данной ситуации. Если генерация наступает, то можно говорить о верной реализации модели, которая правильно отражает поведение реального триггера.

С другой стороны, возможна также следующая реализация модели триггера, которая представлена на рис. 6. И отличие-то, вроде, при этом не очень существенное – просто перенесли установку выходного сигнала непосредственно в цепи, реализующие переходы, устранив в целях экономии две цепи. Но … поведение модели изменилось. Теперь она не входит в режим генерации.

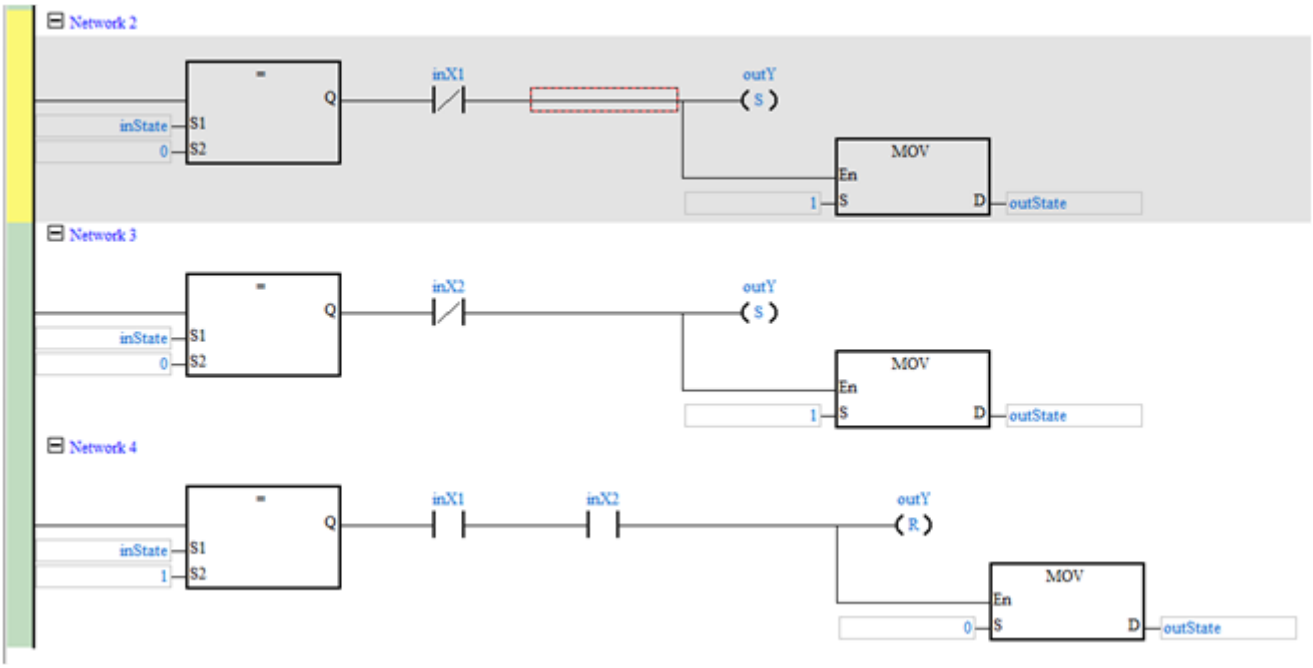


Рис. 6. Измененная модель элемента И-НЕ

В чем дело? А дело в привязке выходных каналов к состояниям модели. В первом случае так мы поступили, создав модель по типу автомата Мура. В ней выходной сигнал изменит свое значение, только при смене текущего состояния. В последнем варианте выходной

сигнал реализован по типу автомата Мили и изменяется раньше - в процессе реализации перехода, т.е. до смены текущего состояния модели (оно на этот момент еще находится в теневом массиве состояний). И именно это фатальным образом сказывается на поведении модели. Но, заметим, если бы был реализован механизм теневой памяти (см. [3]), то поведение модели не зависело бы от выбранного типа автомата.

Рис.7. Реализация простейшего RS-триггера

Наверняка найдутся программисты, у которых автоматное программирование вызывает явное неприятие. И вот только для них мы разработали еще один вариант, который представлен на рис. 7. Глядя на него можно понять их чувства. Но, господа-товарищи, что нам важнее – красивый код или верная его работа? Анализ результатов тестирования приведенных выше двух реализаций придает этому вопросу уже риторическую окраску.

Проектирование первой версии триггера, безусловно, займет больше времени. Но, во-первых, с увеличением сложности создание автоматной модели, наоборот, уменьшает время проектирования. И должно привлекать не время, а качество проектирования. Оно иное, т.к. в противном случае теория автоматов потеряла бы смысл своего существования. Во-вторых, потратив больше времени на разработку модели, мы можем применять ее как компоненту более сложных проектов, будучи уверенными в результатах ее работы. В-третьих, становится возможным качественно иное документирование проектов, т.к. автоматная форма более компактна и информативна, чем блок-схемное сопровождение программных проектов. И в последнем случае нам в помощь будет положительный опыт использования UML [4].

Вместо заключения

Хотелось бы, заключая статью, продолжить жизнеутверждающую тему предыдущего раздела, но ... не получается. А все дело в том, что захотелось получить визуальные подтверждения генерации триггера. И вот с этим-то, как раз, и возникли проблемы... Чтобы однозначно убедиться, что триггер входит в режим генерации, нужно было просто замедлить его работу. И затем по результатам индикации (в нашем случае это светодиоды Y1, Y2) получить наглядное подтверждение, казалось бы, очевидного. Но как это сделать, не изменяя его код?

В ПЛК скорость работы программ зависит от длительностью цикла ПЛК. А он зависит от общего времени последовательной работы программных модулей, находящихся в папке Программы проекта (см. рис.1). Добавляем еще один модуль. В него вставляем уже программный цикл (в ПЛК это будут цепи между командами FOR и NEXT). Смотрим на результаты и ... видим, что к нашему удивлению, светодиоды не одновременно загораются/гаснут, а в некой очередности. При этом, что еще больше поражает, сама генерация протекает, пусть и большими задержками, но исправно.

Не буду утомлять предположениями, версиями и т.п. После вынужденного (в целях выяснения проблемы) создания транспортной/инерционной задержки (подробнее о ней см. [3]), после "камлания" с длительностью цикл в добавленном модуле выяснилось, что, похоже, проблема именно в длительности цикла ПЛК. Но как быть с ней? Тоже не сложно. Избегайте операторов FOR-NEXT. А в ситуации, когда нужен все же программный цикл?! Создайте автоматный алгоритм, как это описано выше. В них эти операторы не нужны и потому длительность цикла работы самого ПЛК не будет фатально от них зависеть.

Т.е. мы разобрались с причиной проблемы, так сказать, вывернулись, т.к. теперь понятно, что и как делать, чтобы ее не было. Другими словами, несмотря ни на что, нормальное заключение, кажется, все же получилось... Благодаря все тем же автоматам. Вот, как все просто! Особенно когда знаешь про автоматы...

Литература

1. Рылов С. Языки программирования стандарта МЭК 61131-3. [Электронный ресурс], Режим доступа: https://finestart.school/media/programming_languages, свободный. Яз. рус. (дата обращения 18.07.2022).

2. Серия AS. Руководство по программированию. [Электронный ресурс], Режим доступа: https://deltronics.ru/images/manual/AS_PrM_RU_[112018].pdf, свободный. Яз. рус. (дата обращения 18.08.2022).

3. АВТОМАТНОЕ ПРОГРАММИРОВАНИЕ: ОПРЕДЕЛЕНИЕ, МОДЕЛЬ, РЕАЛИЗАЦИЯ. https://habr.com/ru/post/682422/

4. Буч Г., Рамбо Дж., Якобсон И. Язык UML. Руководство пользователя. Второе издание. Академия АЙТИ: Москва, 2007. – 493 с.

Теги: автоматное программирование, ПЛК DELTA

Хабы: Параллельное программирование, Промышленное программирование

9 Карма 2 Рейтинг

Вячеслав Любченко @lws0954
Программист

Комментарии 24

Siemargl
28.08.2022 в 23:08

Прекрасный пример, как не надо писать на релейке. Да и заодно убожества Дельты.

Инструкции MOV в релейке быть не должно (разве что не биты а word'ы и прочие данные передать), достаточно койлов, уж тем более, что здесь есть даже их edge-варианты.

Для непривычных, аналогом такого применения MOVв процедурных языках будет if (x=true) y = false

Так же на рис.4 наблюдается нарушения правила единого выхода, что в разных контроллерах - неопределенное поведение (например выход будет моргать по ходу исполнения логики в Allen-Bradley, а в Сименсе не будет)

+4 Ответить

9982th
29.08.2022 в 10:02

> неопределенное поведение (например выход будет моргать по ходу исполнения логики в Allen-Bradley, а в Сименсе не будет)

Как минимум у Сименс это не какая-то особенность реализации, а вполне себе документированная фича с известными правилами работы и целым разделом документации, посвященным ее настройке, поэтому словосочетание "неопределенное поведение" ей, мне кажется, не подходит. Если требуется совместимость с кодом, который завязан на мгновенное обновление, — как минимум в четырехсотых контроллерах эту штуку можно просто отключить в конфигурации.

Однако даже если контроллер будет менять напрямую выходы, перезапись значения в соседней инструкции, вероятнее всего, не будет физически проявляться просто потому, что за время выполнения одной инструкции выход сработать попросту не успеет. Собственно, автору понадобилось создать значительную задержку между выполнением команд чтобы увидеть мигание выходов в такт логике.

0 Ответить

Yukr
29.08.2022 в 10:04

Во-первых, Дельта не убожество, а достаточно хороший продукт по соотношению цена/качество, а также поддержке клиентов. Например, по моему запросу в программе DOPSoft (это дельтовские панели HMI) была изменена кнопка, отображающая последовательность даты/времени в Historic AralmsTrends. Вы скажете - мелочь, я скажу - добейтесь чего-то подобного от Сименса и AB.

Во-вторых, инструкция MOV должна быть, вопрос только в уместном её использовании.

В-третьих, на рис.4 нет нарушения правила единого выхода, но есть ошибка, цепи Network 2 и 3 фактически не будут работать, состояние переменной outState будет определяться цепью Network 4.

Сама статья написана, ИМХО, сложноватым языком. Не корректно использован термин И-НЕ, потому что в качестве "НЕ" использована 3-я переменная. Всё же 2И-НЕ выглядит так:

RS триггер собрается проще, на контактах по переднему/заднему фронту и катушках с SET/RESET функцией. Но это,не зная всю программу, критиковать особо не буду.

Если надо увидеть работу светодиодов - самое простое не замедлять программу, а вставить

Если надо увидеть работу светодиодов – самое простое не замедлять программу, а вставить счётчики, с переменным параметром срабатывания, и отслеживать включение на каждый 5-й,

10-й, ..., 100-й раз.

Самое (для меня) непонятное - а в чём смысл то работы? Паралелльности срабатывания ПЛК всё равно не будет. Если надо одинаковые по времени циклы сканирования - можно отслеживать включение бита в начале цикла, каждый раз его инвертировать, и считать время исполнения, а потом вносить задержку, но это будут миллисекунды.

или как?

0 Ответить

lws0954
29.08.2022 в 10:18

1) Спасибо. Приятно слышать добрые слова в адрес DELT-ы J Плохое ли хорошее, но используем то, что есть. Более того, надеюсь, даже улучшаю. Мне уж точно лучше, чем было (до автоматов, естественно).

2) ПО поводу MOV. В стандарте на LD, кажется, ее нет. Но без нее было бы туго. Если не сказать – невозможно. Но здесь, как мне кажется, нужно воспринимать язык как он есть. Не нравится MOV – не используй. Делов-то!

3) Я не вижу ошибки. Все работает и работает как надо.

По поводу И-НЕ. Речь идет не об операции, а о модели элемента, реализующего данную операцию. Это, скажем так, разные вещи. У Вас все же не 2И-НЕ, а, как я понимаю, 2ИЛИ-НЕ. Но даже не в этом дело. Делают триггер и на этих элементах. Так вот. Если Вы соберете триггер на приведенном Вами элементе, то он работать верно не будет. Это гарантированно. Не верите – попробуйте и сами убедитесь. Речь прежде всего идет о генерации. Ее не будет 100%. Устойчивые же состояния будут отражаться.

Возможно, сам триггер можно собрать и проще (хотелось бы увидеть как). Но смысл в том, чтобы избегать каких-то хитрых приемов, упрощающих что-то. Ну и потом убедиться еще надо, что он будет работать правильно. Т.е. как настоящий триггер, а не просто как нечто, имеющее два устойчивых состояния. Это будет «утка», которая крякает, но не плавает J

О замедлении. В том-то и дело, что хотелось бы замедлить программу, не вставляя в нее что-то. Так обеспечивается чистота эксперимента. И это первое что пришло в голову. Т.е. чтобы совсем не касаться модуля PRG RS_триггер. Но можно создать счетчики и «врезать» их после выходных каналов элементов. Это корректно. Так в конечном итоге и пришлось сделать. Только у меня они названы более точно – задержками. А эти задержки могут или транспортными или инерционными. Но это уже более детальный разговор. И сделаны они должны быть тоже, как автоматы. Если вставлять простые счетчики, то придем к эффекту, который случился и у меня.

И о самом непонятном (или непонятом?). О смысле работы. Он вроде бы озвучен в начале статьи – в реализации идеи. Идеи автоматного программирования (прошу прощения, что не прописал прямо, но рассчитывал все понятно и так). Или, что точнее (но по смыслу будет уже), реализации автоматов на языке LD. И именно это (реализация идеи) и только это создаст

параллельность, которой в исходном варианте программирования нет и в помине.

0 Ответить

Yukr
29.08.2022 в 10:41

прошу простить, не тот скриншот прикрепил для 2И-НЕ, вот этот правильный

0 Ответить

lws0954
29.08.2022 в 10:55

Так и у меня тот же на рис.7. Просто там их два штуки, как и нужно для триггера.

0 Ответить

Yukr
29.08.2022 в 11:44

видимо, я недопонимаю Вашу идею автоматов. Прочитал Вашу статью из списка литературы. Сложновато для меня, но попробую разобраться))

0 Ответить

Astronom71
29.08.2022 в 09:59

Какое-то странное восприятие PLC - как слабый ПК и с убогими языками программирования. 25 лет в автоматизации, но такие ассоциации первый раз прочел, спасибо за расширение моего понимания, тем чем я всю жизнь занимаюсь

0 Ответить

lws0954
29.08.2022 в 10:56

Да, как говорится, не за что ;) Нужно смотреть объективно. Поскольку я до этого программировал на C++ и на обычным ПК, то сравнивать есть с чем. Так что, не в обиду, но возможностей много-много меньше. Но это не значит, что совсем хуже. В чем-то даже лучше и проще. Я и об этом сказал. Но, например, свой код на C++ (аналог его) на ПЛК я уже не перенесу. В принципе. Но что-то получилось (в смысле идеи, но не кода). И во многом об этом

статья.

0 Ответить

Yukr
29.08.2022 в 12:11

Возможно, Вам больше подошел бы язык ST, он ближе к C, вот тут первоначальные сведения есть. http://www.codesys.ru/docs/st_c.pdf

Правда ST не доступен на контроллерах DVP, только на AS200-300. И на нём сложно делать импульсные цепи. Но мультивибратор я как то делал.

0 Ответить

Astronom71
29.08.2022 в 13:29

Вы не поняли мой стёб, то что вы пришли с C++, я как раз и понял после первых абзацев вашей статьи. Мне немного не понравилось как вы описали контроллер. В моем понимании (без смотрения в Вики) это комплекс аппаратных и программных средств обеспечивающие управление тех. процессом в реальном времени. Быстро с головы написал, надеюсь асушники меня какашками не забросают

p.s. ну и все же нужно не подыскивать удобный язык программирования а прочесть основы - **Ганс Бергер** это для Siemens контроллеров, но я думаю можно найти и литературу не специализированную к железу. И тогда проблем с языками не будет.

p.p.s. - Соответственно не будет в логике множественных присвоений

0 Ответить

lws0954
29.08.2022 в 16:29

Может, это опять стеб, но проблем с языками у меня лично нет. Просто LD был почти один доступный язык на момент начала работы с данным типом ПЛК.

И по поводу "множественности". Их не много и не мало, а столько сколько нужно при выбранном способе реализации автоматов. Если Вы сможете предложить какой-то свой вариант, уменьшающий их число, то можно будет обсудить и сравнить.

И обращу внимание еще раз. мы боремся не за красивый или минимальный код, а за такой, который отражает просто, однозначно, наглядно и надежно модель программирования. В данном случае - автоматного. Здесь каждая цепь - это один переход (одна дуга) автомата. Поэтому зачем "портить" то, что и так хорошо ;)

0 Ответить

Astronom71
29.08.2022 в 16:46

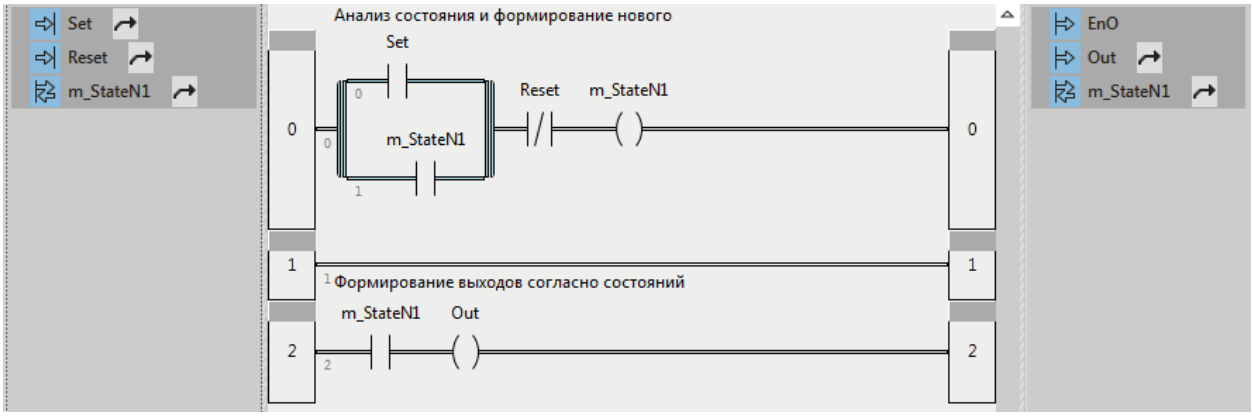
Не вижу весь код, потому могу судить не корректно, если outstate у вас аналог состояния графа, то да, все норм)

0 Ответить

Siemargl
29.08.2022 в 19:14

Все там видно на рис.4 - бардак.

Типичный RS-триггер из любого учебника по электротехнике - схема управления двигателем



Для алгоритмов посложнее добавить количество m_StateXXX (ес-но с хранением состояния)

0 Ответить

lws0954
29.08.2022 в 22:44

> Все там видно на рис.4 - бардак.

Бардак - это одна из форм порядка :)

Потом, бардак это для непросвещенных. А так - полный порядок. Но порядок здесь в наличии модели автомата, представляющей алгоритм программы. Я, например, даже в своих программах на LD не разбираюсь, когда возникают проблемы. Я смотрю на автомат, изменяю его и потом корректирую программу на LD. А без этого уже спустя день-второй я тоже смотрю на свою LD-программу и думаю - какой же там бардак :)

> Типичный RS-триггер ...

Прошу прощения, но что-то я в Вашем коде не увидел "типичного триггера". Может можно представить Вашу идею (алгоритм) в форме ... автомата. Ну, на крайний случай, в форме блок-схемы. Если, конечно, это возможно. Но это просто должно быть возможно! Если, конечно, Ваш код не одна из форм настоящего бардака, для которого это сложно сделать (саму невозможность я исключаю совсем) ;)

0 Ответить

Siemargl
30.08.2022 в 00:29

> Прошу прощения, но что-то я в Вашем коде не увидел "типичного триггера"

> Прошу прощения, но что-то я в Вашем коде не увидел "типичного триггера"

Без проблем, 1я же картинка.
https://studref.com/354825/stroitelstvo/tipovye_uzly_shemy_upravleniya_elektroprivodov_asinhronnymi_dvigatelyami

0 Ответить

lws0954
30.08.2022 в 15:19

Триггера там, конечно, нет. Но есть штука похожая на него ;) А вот и автоматная модель, аналогичная схеме на рис. 3.8. и которую Вы, к сожалению, так не создали, может быть следующей:

Рис_3_8 **POU(Движок)**
awStates[10]

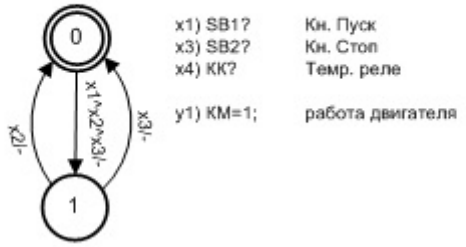
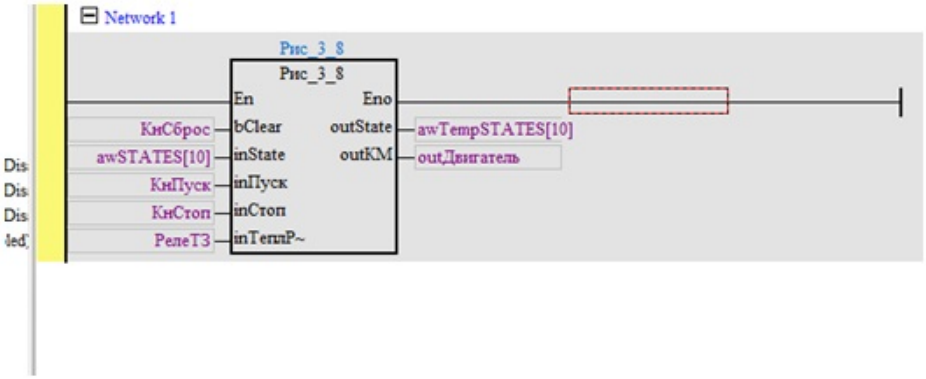


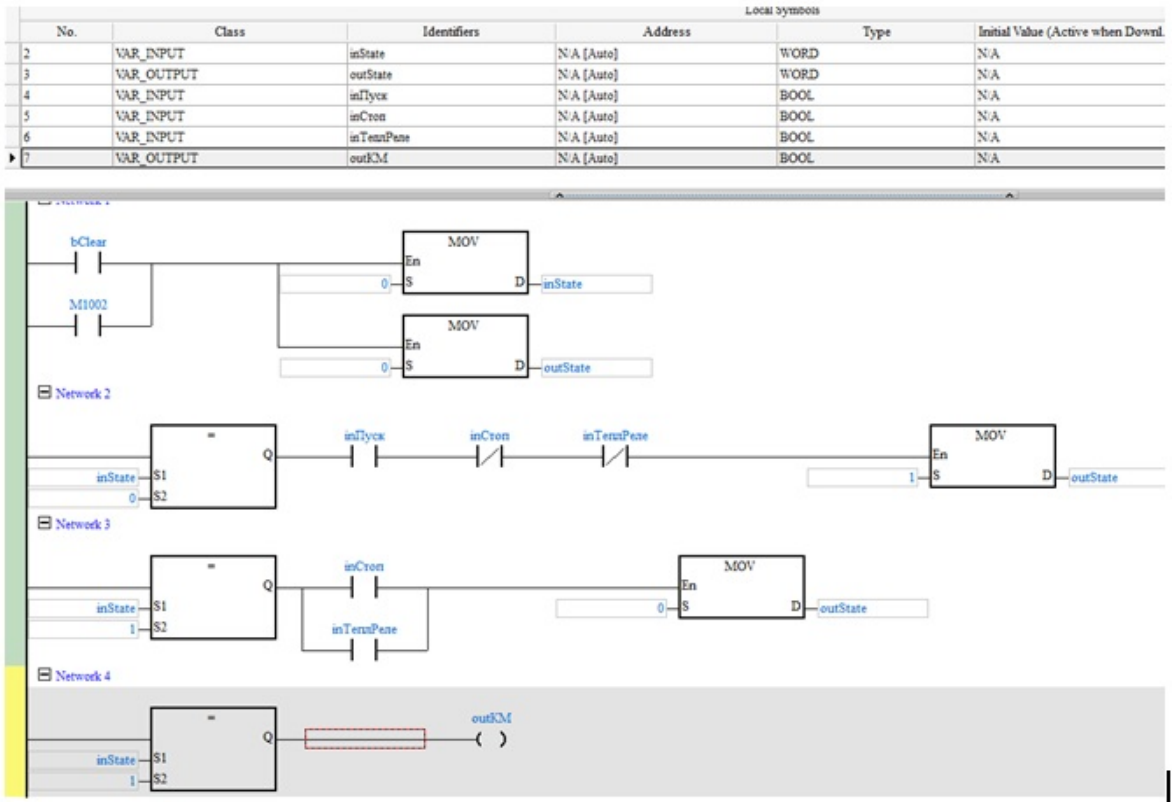
Рис. Алгоритм управления двигателем с использованием нереверсивного магнитного пускателя

Краткое описание алгоритма:

Для пуска двигателя нажимают кнопку пуска SB1. Получает питание катушка контактора КМ, который, включившись, своими главными силовыми контактами в цепи статора двигателя подключает его к источнику питания. Происходит разбег двигателя. Для отключения двигателя нажимается кнопка остановки SB2, контактор КМ теряет питание и отключает двигатель от сети (двигатель также отключается при срабатывании температурного реле КК). Начинается процесс торможения двигателя.

Программа и ФБ на языке LD, которые соответствует данному автомату, будут такие:



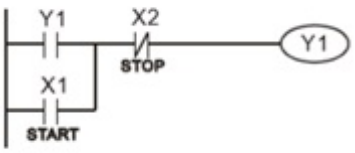


Это полный код. Он, конечно, побольше, чем у Вас. Но, во-первых, ему я доверяю: автомат достаточно прост, а перевод на язык LD снадартен и проверен. А вот Вашу Программу еще нужно тестировать на соответствие схеме.

Но я вспомнил, что где-то видел аналог Вашего “триггера», и даже нашел его. Это в документации по программированию на ПЛК Дельта. Но называется там эта схема – "Самоблокировка выхода с приоритетом Стопа" и, наверное, так будет точнее:

Пример 1. Самоблокировка выхода с приоритетом Стопа

При нажатии кнопки X1=On, сигнал проходит через нормально замкнутую кнопку X2 и вызывает замыкание катушки Y1. При этом замыкается связанный входной контакт Y1. При нажатии кнопки X2 (Стоп) цепь разомкнется и катушка (выход) Y1 отключится. Поэтому данную схему называют приоритетом Стопа.



Но приведена там же и, так называемая, триггерная схема (Пример 10):

Но это тоже не триггер, конечно.

0 Ответить

lumen_xp
29.08.2022 в 10:04

Почему-то программисты ПЛК забывают, что контроллер промышленный это дискретная штука. В Сименсе моргать выходы не будут, потому что они устанавливаются в начале цикла ПЛК, потом идет чтение входов, выполнение организационных блоков и т.д.

В AB скорее всего сделали иначе и состояние выходов меняют по ходу выполнения программы, что не очень то логично, RLO хотелось бы видеть в конце цикла, а не как программисту в голову взбредет.

В общем то есть детские ошибки, которые лечатся нормальной пусконаладкой и выдиранием волос из мягкого места спины, когда ничего не работает, а заказчик требует и угрожает расправой.

По поводу Дельты, это нормальная альтернатива в современных условиях для систем среднего класса. Но у них тоже сроки поставки конские.

0 Ответить

Sergeant101
29.08.2022 в 10:43

Изменение выходов, а равно как и чтение входов по ходу выполнения программы нарушает пожалуй самый главный из принципов - детерминированного поведения: при выполнении цикла программы входа\выхода не меняются, дабы не получить трудноуловимый undefined behavior.

0 Ответить

Yukr
29.08.2022 в 11:41

не уверен насчёт AB, но в Дельте цикл такой: Чтение ВХОДОВ -> работа программы слева направо и сверху вниз по цепям -> запись ВЫХОДОВ

0 Ответить

lws0954
29.08.2022 в 13:58

Все это так. Но что понимать под циклом программы. У автоматов цикл - это переход между

состояниями, т.е. один такт дискретного времени. В программах для ПЛК, судя по описанию, это работа программы от начала и до оператора END (если несколько программ, то сумма времен). А если в программе будет цикл. Да, не дай Бог, еще и бесконечный? Да, в ПЛК превышение цикла контролируется. Но если цикл нужен? У автоматов таких ограничений нет. Контролируется только такт дискретного времени. Автоматная программа выполняет только один шаг – переход из текущего состояния и после этого передает управление другому автомату. И в сумме эти шаги не должны превышать заданную величину дискретного такта. Это в синхронном режиме, т.е. с фиксированным значением дискретного такта. Но эта величина может быть и плавающей – режим асинхронной работы автоматов.

0 Ответить

Siemargl
29.08.2022 в 18:22

Вообще автоматом является управляемый объект, и его состояния (наливается, загружается, перемешивается...), а что там в программе происходит - дело пятое и уже тем более уровень программных инструкций не интересен.

0 Ответить

lws0954
30.08.2022 в 15:49

Вообще-то речь о других автоматах. О математической модели так называемого преобразователя информации (см. Введение в кибернетику. Глушков В.М.). Да и любая книга на тему конечных автоматов поясняет, о каких таких автоматах идет здесь речь. Наш контекст - конечные автоматы, как формальная алгоритмическая модель для объектов, реализующих функции управления в форме [любых] алгоритмов. Это ПК, ПЛК и т.п. В процессе могут создаваться и модели объектов управления. Нет проблем. Иногда это очень даже нужно (например, для адаптивного управления). И здесь уже нужно подкорректировать свое понимание понятия "состояния". Оно может отождествляться с теми процессами, что упомянули Вы (наливай и т.п.) , но совсем не являться ими.

0 Ответить

lws0954
30.08.2022 в 15:56

> По поводу Дельты, это нормальная альтернатива в современных условиях для систем среднего класса. Но у них тоже сроки поставки конские.

Все так. Но заменить чем-то не очень реально. Остается ждать и надеяться, т.к. с другой движухой - в сторону замещения как-то не очень... Печально, но факты вещь упрямая.

0 Ответить

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

ПОХОЖИЕ ПУБЛИКАЦИИ

14 августа в 18:08

Автоматное программирование: определение, модель, реализация

+2 2.5K 32 39 +39

18 ноября 2017 в 06:56

Автоматное программирование. Часть 4. Эффективность автоматно-спроектированных программ

+10 8.7K 48 11 +11

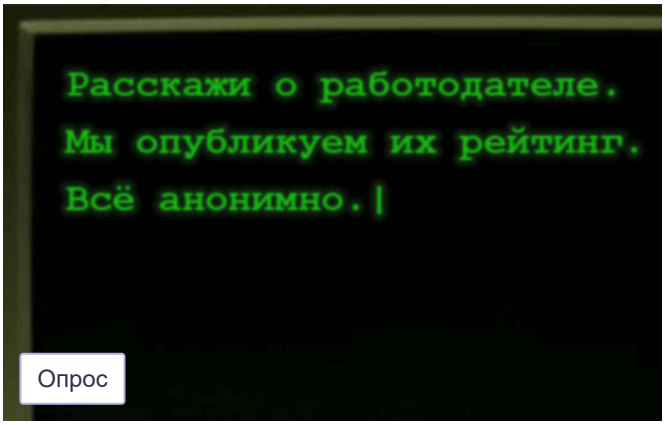
11 ноября 2017 в 07:21

Автоматное программирование. Часть 3. Диаграмма состояний и переходов. Продолжение

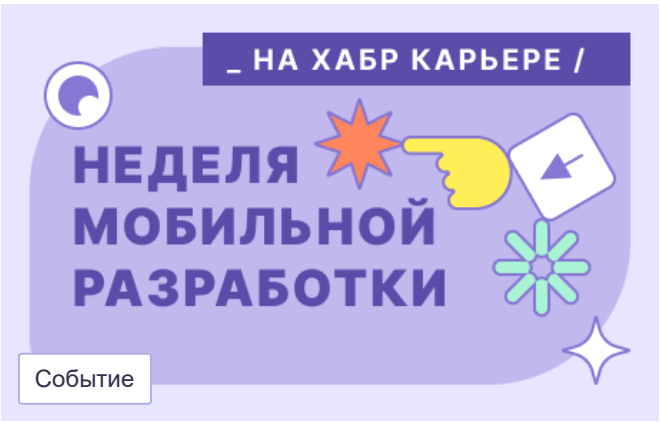
+8 19K 84 20 +20

МИНУТОЧКУ ВНИМАНИЯ

Разместить



Третье хабраисследование ru-IT-брендов

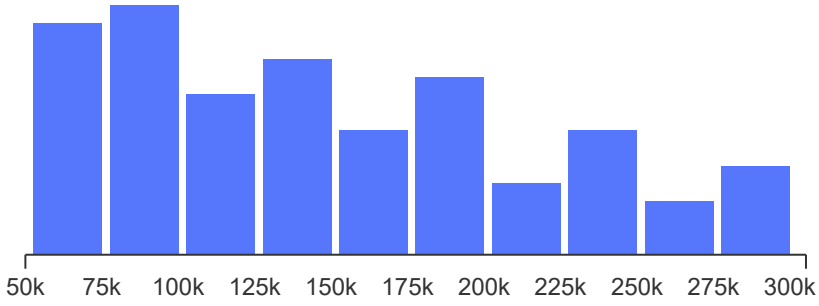


Неделя мобильных разработчиков на Хабр Карьере

СРЕДНЯЯ ЗАРПЛАТА В IT

158 104 ₽/мес.

— средняя зарплата во всех IT-специализациях по данным из 4 490 анкет, за 2-ое пол. 2022 года. Проверьте «в рынке» ли ваша зарплата или нет!



Проверить свою зарплату

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

сегодня в 10:01

Что случается с металлоломом и зачем там хардкорное ИТ

+126 6.2K 20 33 +33

сегодня в 04:39

У вас captcha? Тогда мы уходим от вас

+45 16K 19 130 +130

сегодня в 11:00

Мифы и легенды современного Python

+30 5K 32 19 +19

сегодня в 10:00

Механизмы — хорошо, а олени — лучше, или Техсервис в условиях климатического экстрима

+26 921 13 0

сегодня в 12:00

Правим QEMU железным кулаком

+23 2.4K 27 11 +11

Налево пойдешь — в digital попадешь, направо покатишься — в IT вкатишься, а тут кликнешь — и в опрос залипнешь

Опрос

ЧИТАЮТ СЕЙЧАС

Как я нахожу парковочное место за 5 секунд

57K 246 +246

Реверс-хантинг

2.5K 11 +11

Импортозамещение по-американски: крупнейшие производители полупроводников мира строят фабрики в США

4.5K 11 +11

Как перестать быть сутулой собакой: мой путь к здоровой спине

5.5K 44 +44

У вас captcha? Тогда мы уходим от вас

16K 130 +130

DAST ist fantastisch: отечественный динамический анализатор к взлету готов

Мегапост