Инженерный вестник Дона, №2 (2025)
ivdon.ru/ru/magazine/archive/n2y2025/9840

Методы решения проблемы замедления рабочего цикла контроллера при увеличении объёма программы

А.Н. Долидзе

Государственный университет аэрокосмического приборостроения, Санкт-Петербург

Аннотация: В статье рассматриваются варианты решения проблемы замедления цикла работы программируемых логических контроллеров, возникающей при реализации сложных алгоритмов. У большинства современных программируемых логических контроллеров цикл работы напрямую зависит от объёма пользовательской программы. Программа выполняется псевдопараллельно, поэтому сложность алгоритма влияет на замедление цикла косвенно, через увеличение объёма кода. Разрастание программы приводит к замедлению отклика контроллера на изменения состояния входов. В некоторых случаях разработчики контроллеров предоставляют программистам возможности, позволяющие нарушить естественный порядок выполнения рабочего цикла, тем самым сокращая время реакции. Достоинства и недостатки таких методов подробно разобраны в данной статье. В качестве альтернативного способа уменьшения времени реакции контроллера рассматривается возможность передачи исполнения части алгоритма сенсорной панели оператора. Современные сенсорные панели, помимо своей основной задачи – реализации человеко-машинного интерфейса, обладают множеством дополнительных функций, среди которых возможность применения макросов. Приведён основной функционал макросов и продемонстрирована возможность обмена данными между контроллером и панелью. Эта возможность является необходимым условием для делегирования части функций контролера панели. Подробно разобраны ограничения и риски, возникающие при применении данного подхода, а также обозначены ситуации, в которых предпочтительно применять этот метод.
Ключевые слова: программируемый логический контроллер, сенсорная панель оператора, цикл работы контроллера, распределение вычислительных ресурсов, макросы.

Введение

Одной из характерных особенностей программируемых логических контроллеров (ПЛК) является их цикл работы [1], который, укрупнённо, сводится к трём шагам: считывание состояния входов, исполнение программы, изменение состояние выходов (рис.1). На этапе считывания состояния входов их значение фиксируется в специальных регистрах (регистрах образов входов). Во время исполнения программы однократно выполняется каждая её инструкция, при этом в качестве исходных данных используется не текущее состояние входов контроллера, а регистр образов входов. Выходные сигналы, формируемые в процессе выполнения

программы, также не отправляются непосредственно на выход контроллера, вместо этого они фиксируются в соответствующих специальных регистрах (регистрах образов выходов). После того как все команды программы будут обработаны, контроллер переходит к изменению состояния выходов, только в этот момент данные из регистров образов выходов переносятся на реальные выходы ПЛК.

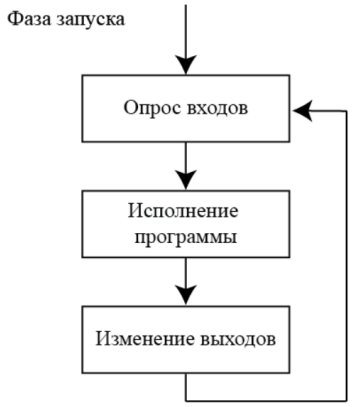

Рис.1. – Цикл работы ПЛК

Основной недостаток такого подхода довольно очевиден: чем больше и сложнее пользовательская программа, тем реже опрашивается реальное состояние входов, и дольше формируются выходные сигналы (рис.2).

Рис.2. – Влияние объёма программы на цикл работы ПЛК: а – небольшая программа; б – задержка реакции на событие, в результате увеличения объёма программы

Стадийное программирование

Существуют различные методы борьбы с этой проблемой, например – стадийное программирование [2]. При стадийном программировании тело программы разбивается на ряд стадий, причём каждая имеет свои условия выполнения. Контроллер по-прежнему работает в стандартном цикле, но на этапе выполнения программы он игнорирует те стадии, условия запуска которых не выполнены, таким образом время обработки программы значительно сокращается, а скорость реакции на события увеличивается. Разумеется, применение стадийного программирования возможно только если разработчик ПЛК предоставил соответствующие инструменты прикладному программисту, к сожалению, такая возможность доступна далеко не для каждого контроллера. При использовании стадийного программирования необходимо понимать, что данный подход может породить существенную проблему: программа может задержаться в конкретной стадии на продолжительное время, если ее разбиение продуманно недостаточно тщательно, при этом остальные стадии будут проигнорированы контроллером, из-за чего события, обрабатываемые ими, могут быть пропущены или утратить актуальность.

Непосредственный доступ к входам и выходам ПЛК

Иногда, в качестве еще одного метода решения описанной проблемы, разработчики ПЛК предоставляют функции для непосредственного доступа к входам и выходам ПЛК, например, для контроллеров Mitsubishi серии FX, предусмотрена команда REF (рис.3), позволяющая обновить выбранные биты регистров образов входов или выходов [3].

Рис. 3. – Применение команды REF (язык LD) для обновления информации о состоянии входа X000

Как следует из названия, эти функции, в момент своего исполнения, игнорируют порядок цикла работы ПЛК и обращаются непосредственно к реальному входу или выходу контроллера, минуя регистры образов, в результате чего происходит мгновенная реакция на события. При использовании подобных функций неизбежно возникают некоторые накладные издержки, связанные с нарушением цикла работы и вызывающие задержку исполнения остальной программы [4]. Подобные функции хорошо справляются с проблемой, но, как и стадийное программирование, поддерживаются далеко не каждым ПЛК.

Делегирование вычислений сенсорной панели оператора

В статье [5] среди методов решения проблемы нехватки входных/выходных сигналов ПЛК описывался подход с использованием сенсорной панели оператора, это же решение может помочь сократить время рабочего цикла контроллера. Сенсорная панель оператора может быть достаточно сложным устройством, оснащённым полноценным микропроцессором и внутренней памятью, а также набором портов для подключения внешних устройств [6]. В таком случае функционал панели оператора, помимо основной задачи – реализации человеко-машинного интерфейса (ЧМИ), расширяется множеством дополнительных функций. Одна из таких функций: возможность использования макросов – подпрограмм, вызываемых различными событиями, как внутренними, относительно панели, так и внешними. Такие подпрограммы могут взять на себя реализацию некоторых функций основной программы ПЛК, тем самым сократив время выполнения его рабочего цикла (путём уменьшения объёма основной программы).

Рассмотрим в качестве примера серию сенсорных панелей оператора MT8000 фирмы Weintek [7]. Итак, в данном случае, макрос – программа, написанная на специальном языке программирования высокого уровня,

разработанном специально для панелей Weintek. Данный язык, как и большинство языков высокого уровня, поддерживает выполнение циклов и условных операторов, что позволяет разрабатывать довольно сложные алгоритмы. Операторы языка позволяют использовать логические, математические и тригонометрические функции, представленные в таблице №1.

Таблица № 1

Некоторые функции языка макросов Weintek

Функция	Описание
=	оператор присваивания
+, -, *, /, %	арифметические операторы
And, Or, Xor, Not	операторы, применяемые для построения логических выражений
<<, >>	операторы сдвига
&, |, ^	побитовые операции
SQRT	извлечение квадратного корня
SIN, COS, TAN, COT, SEC, CSC, ASIN, ACOS, ATAN	тригонометрические функции
RAND	получение псевдослучайного числа

Помимо представленных в таблице №1, существует ещё множество полезных функций, например, различные варианты преобразования формата кодировки переменных (двоичный, двоично-десятичный, десятичный, шестнадцатеричный, ASCII) или изменения порядка расположения байтов в

слове. Здесь стоит отметить, что далеко не каждый контроллер поддерживает перечисленные функции, например LOGO! [8] фирмы Siemens поддерживает только базовые арифметические операции, описанные в статье [9]. Таким образом, панель оператора может значительно расширить функционал основного контроллера в системе управления, внеся дополнительные возможности, не предусмотренные стандартными языками для ПЛК [10].

Ключевым моментом для использования макроса в поставленной задаче является возможность обмена данными с ПЛК. Панели необходимо получать исходные данные от контроллера (состояние входов, выходов и внутренних переменных), а также передавать обратно результаты своих вычислений. Макросы панели оперируют своими внутренними переменными, которые не являются регистрами памяти панели или контроллера, память для них временно выделяется в оперативной памяти панели на момент выполнения макроса. За обмен данными между переменными макроса и регистрами панели или контроллера отвечают функции GetData (получить данные) и SetData (передать данные). На рис.4 представлен макрос, копирующий состояние первого входа LOGO! в булеву переменную *a*.

```
1  //Запись состояния первого входа ПР LOGO! в переменную a
2  macro_command main()
3  bool a
4  GetData(a, "LOGO", I, 1, 1)
5  end macro_command
```

Рис. 4. – Макрос для переноса состояния первого входа контроллера LOGO! во внутреннюю переменную a

Аргументами команды GetData являются: *a* – имя внутренней переменной макроса, в которую будет производиться запись; “LOGO” – имя

устройства, с которым будет происходить обмен данными; *I* – область памяти устройства, к которой происходит обращение (в данном случае регистр «образа входов»); *1* – адрес смещения от начала области памяти (первый вход контроллера); *1* – номер контроллера в рамках проекта панели. Команда SetData используется аналогично GetData, обе команды могут работать как с одной переменной, так и с массивами данных.

Следующим ключевым моментом является запуск макроса. В зависимости от решаемой задачи существует несколько вариантов запуска макроса. Если макрос необходим для инициализации системы после запуска, то можно использовать однократное выполнение при запуске панели. В случае, когда необходимо периодическое выполнение – можно настроить интервал автоматического запуска. Описанные варианты запуска настраиваются путём установки соответствующего флага в редакторе макросов (рис.5).

Рис. 5. – Фрагмент окна редактора макросов, позволяющего настроить автоматическое выполнение

Нужно понимать, частое выполнение макроса может негативно сказаться на работе панели оператора (если макрос сложный, он замедлит работу панели) или контроллера (при многократном обращении к регистрам контроллера, время рабочего цикла ПЛК увеличится из-за дополнительных операций передачи данных). Желательно вызывать макросы событийно, тогда, когда их результат действительно необходим. При настройке условного выполнения можно выбрать внутренний бит памяти панели, при

установке которого будет выполняться макрос. Этот способ удобен, если запуск макроса будет производиться путём нажатия кнопки на панели, но для автоматического запуска он сложен. Основное устройство в системе управления – контроллер, именно в нём находится программа, реализующая логику работы, а значит, события, вызываемые контроллером, должны запускать макросы. Можно синхронизировать переключение битов в регистрах панели и контроллера, но это косвенный метод, сопровождающийся бессмысленным расходом памяти панели. Более разумной альтернативой упомянутому подходу является использование функции «PLC control» («Триггеры ПЛК» в русифицированной версии среды разработки). С помощью этой функции можно настроить различные события, вызываемые изменением состояния тех или иных областей памяти панели и контроллера, например, на рис.6 показана настройка запуска макроса при изменении флага M1 контроллера LOGO!.

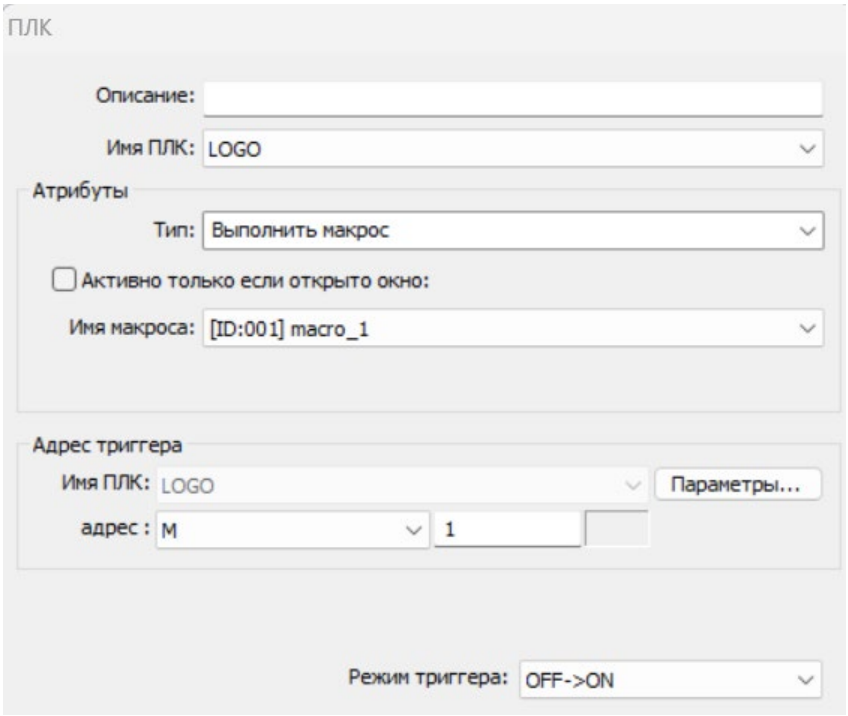

Рис. 6. – Настройка запуска макроса при установке флага M1 контроллера LOGO!

Описанный подход позволяет контроллеру запускать макросы напрямую и без вмешательства оператора. Фактически, программист получает возможность организовать полноценное стадийное программирование даже на том контроллере, для которого оно не предусмотрено (программа контроллера будет определять, какую стадию нужно запустить сейчас, а сами стадии будут выполняться панелью, не перегружая контроллер работой).

Риски и ограничения при использовании макросов панели оператора

Несмотря на обозначенные преимущества, делегирование части вычислительных функций накладывает некоторые ограничения и вносит дополнительные риски. Количество доступных в проекте макросов не бесконечно, например, для серии панелей MT8000 их может быть не более 256. Объём оперативной памяти под внутренние переменные также ограничен, в данном случае можно использовать максимум 4 килобайта. Макросы могут привести к зависанию панели (например, в результате обработки бесконечного цикла), что в свою очередь может привести к нарушению работы всей системы управления (если панели будут поручены ответственные функции). Распределение вычислительных нагрузок также опасно при ненадёжном физическом подключении устройств, обрыв линии связи, очевидно, прервёт функционирование системы. Выше было сказано о том, что процесс передачи данных может негативно сказаться на работе обоих устройств, и, в худшем случае, полностью нивелировать выигрыш от распределения ресурсов. Выполнение макроса панелью занимает некоторое время, иногда эта задержка может стать критической и привести к ошибкам в работе системы.

Выводы

Исходя из перечисленных рисков и недостатков, предложенным методом нужно пользоваться осмотрительно и лишь в тех случаях, когда он может предоставить существенное преимущество, а именно: если памяти контроллера не хватает для полноценной реализации алгоритма или необходимо расширить возможности контроллера сложными функциями. Также метод эффективен, если требуется реализовать стадийное программирование на тех контроллерах, для которых оно не предусмотрено. Уменьшение объёма основной программы контроллера, за счёт распределения вычислительных ресурсов, может увеличить частоту опроса входов и ускорить реакцию выходов, но добиться ощутимых результатов можно только в том случае, если панель будет исполнять наименее критичные к скорости реакции и, по возможности, автономные части алгоритма (например: осуществление контроля медленных процессов, таких как нагревание резервуара с жидкостью).

Литература

1. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. Москва: Солон-Пресс, 2016. 254 с.

2. Programming Styleguide for S7-1200/1500. Nuremberg: Siemens AG, 2015. 109 p.

3. Серия MELSEC FX. Программируемые логические контроллеры FX1S, FX1N, FX2N, FX2NC, FX3U. Токио: Mitsubishi Electric, 2008. 702 с.

4. Hugh Jack Automating Manufacturing Systems with PLCs. Lulu.com, 2010. 644 с.

5. Долидзе А.Н. Методы решения проблемы нехватки входов и выходов контроллера при разработке автоматизированных систем управления // Инженерный вестник Дона, 2024, №9. URL: ivdon.ru/ru/magazine/archive/n9y2024/9500.

6. Ивлев П. Панель оператора как средство создания высокоэффективного HMI // Control Engineering Россия. 2018. №2. С. 41-46.

7. MT-8000/6000 series. Human Machine Interface with 7" TFT LCD display // Taipei: Weintek, 2014. 8 p.

8. Siemens AG LOGO! - System Manual. Nuremberg: Siemens AG, 2022. 371 p.

9. Долидзе А.Н. Обзор специфических функций языка FBD на примере программируемых реле LOGO! // Инженерный вестник Дона, 2022, №11. URL: ivdon.ru/ru/magazine/archive/n11y2022/7991.

10. Programmable controllers – Part 3: Programming languages / International Electrotechnical Commission, International Electrotechnical Commission, 2013. 438 p.

References

1. Petrov I.V. Programmiruemye kontrollery. Standartnye yazyki i priyomy prikladnogo programmirovaniya [Programmable controllers. Standard languages and techniques of application programming]. Moskva: SOLON-Press, 2016. 254 p.

2. Programming Styleguide for S7-1200/1500. Nuremberg: Siemens AG, 2015. 109 p.

3. Seriya MELSEC FX. Programmiruemye logicheskie kontrollery FX1S, FX1N, FX2N, FX2NC, FX3U [MELSEC FX series. Programmable logic controllers FX1S, FX1N, FX2N, FX2NC, FX3U]. Tokyo. Mitsubishi Electric, 2008. 702 p.

4. Hugh Jack Automating Manufacturing Systems with PLCs. Lulu.com, 2010. 644 p.

5. Dolidze A.N. Inzhenernyj vestnik Dona, 2024, №9. URL: ivdon.ru/ru/magazine/archive/n9y2024/9500.

6. Ivlev P. Control Engineering Rossiya. 2018. №2. pp. 41-46.

7. MT-8000/6000 series. Human Machine Interface with 7" TFT LCD display. Taipei: Weintek, 2014. 8 p.

8. Siemens AG LOGO! - System Manual. Nuremberg: Siemens AG, 2022. 371 p.

9. Dolidze A.N. Inzhenernyj vestnik Dona, 2022, №11. URL: ivdon.ru/ru/magazine/archive/n11y2022/7991.

10. Programmable controllers – Part 3: Programming languages. International Electrotechnical Commission, International Electrotechnical Commission, 2013. 438 p.

Дата поступления: 21.12.2024

Дата публикации: 27.01.2025