МЭК 61131-3

Взгляд со стороны программистов ПЛК

14.02.2022
версия 1.1

https://oscat.ru/

Оглавление

Введение

Первый программируемый логический контроллер (ПЛК) – Modicon 084 – был представлен компанией Modicon (созданной коллективом инженеров компании Bedford Associates) в 1969 году[1]. Это устройство было разработано в рамках тендера, объявленного компанией General Motors. Целью General Motors было заменить громоздкие и сложные в обслуживании шкафы релейной автоматики на компактные вычислительные устройства. Поскольку настраивать эти устройства должны были те же самые техники, которые до этого собирали релейные схемы – разработчики Modicon 084 создали для них язык программирования, соответствующий их потребностям – язык релейных диаграмм, который позже получит название LD.

Постепенно к производству контроллеров приступили и другие компании. Некоторые из них стали разрабатывать свои языки программирования – часто совершенно непохожие и несовместимые между собой. Это естественным образом привело к необходимости стандартизации языков программирования ПЛК. Разработка такого стандарта началась в 1979 году[2], а первая официальная версия была выпущена в 1993. Стандарт получил обозначение IEC 61131-3 (МЭК 61131-3). Он описывает 5 языков программирования ПЛК:

- LD – графический язык, основанный на релейных диаграммах;
- FBD – графический язык, основанный на функциональных блоках;
- SFC – графический язык, основанный на диаграммах состояний;
- IL – текстовый ассемблер-подобный язык;
- ST – текстовый язык, основанный на структуре и синтаксисе языка Pascal.

В 2003 году вышла вторая редакция стандарта, принесшая ряд точечных дополнений и исправлений. В 2013 была опубликована третья редакция, основным нововведением которой стала поддержка объектно-ориентированного подхода (ООП). В данный момент идет работа над четвертой редакцией, которая ориентировочно должна быть опубликована во второй половине 2024 года.

Наличие в стандарте пяти языков вызывает интересный вопрос – насколько предпочтительными с точки зрения инженеров-программистов являются те или иные из них? Понятно, что некоторые производители поддерживают для своих ПЛК только один или два языка, не оставляя особого выбора. Но современные среды разработки крупных производителей (TIA Portal, GX IEC Developer, PLCnext Engineer, CODESYS, ISaGRAF и т.д.) позволяют использовать все 5 языков стандарта МЭК.

Опросов по сравнительной популярности языков программирования ПЛК не так уж и много. Ниже мы приведем результаты тех из них, которые нам удалось найти.

[1] Ванесса Ромеро Сеговия, Альфред Теорин. История управления. История ПЛК и РСУ
[2] Karl Heinz JohnMichael Tiegelkamp. IEC 61131-3: Programming Industrial Automation Systems

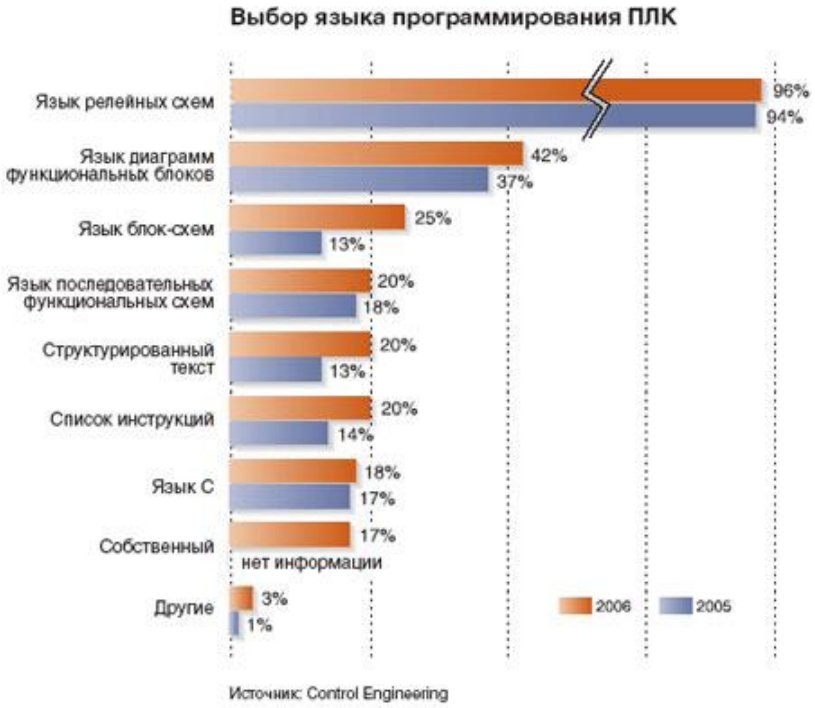

Рис. 1. Опрос журнала Control Engineering (2006)

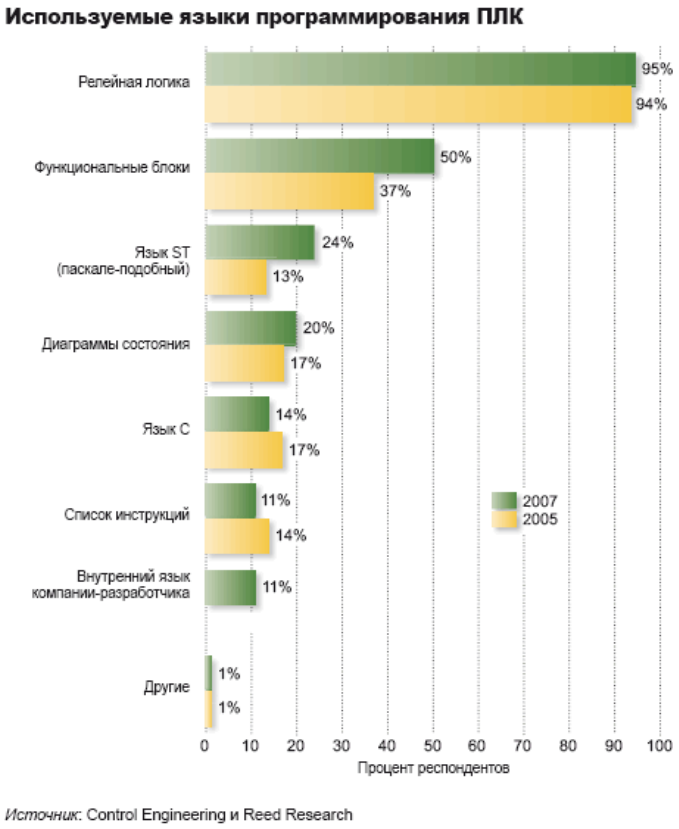

Рис. 2. Опрос журнала Control Engineering (2007)

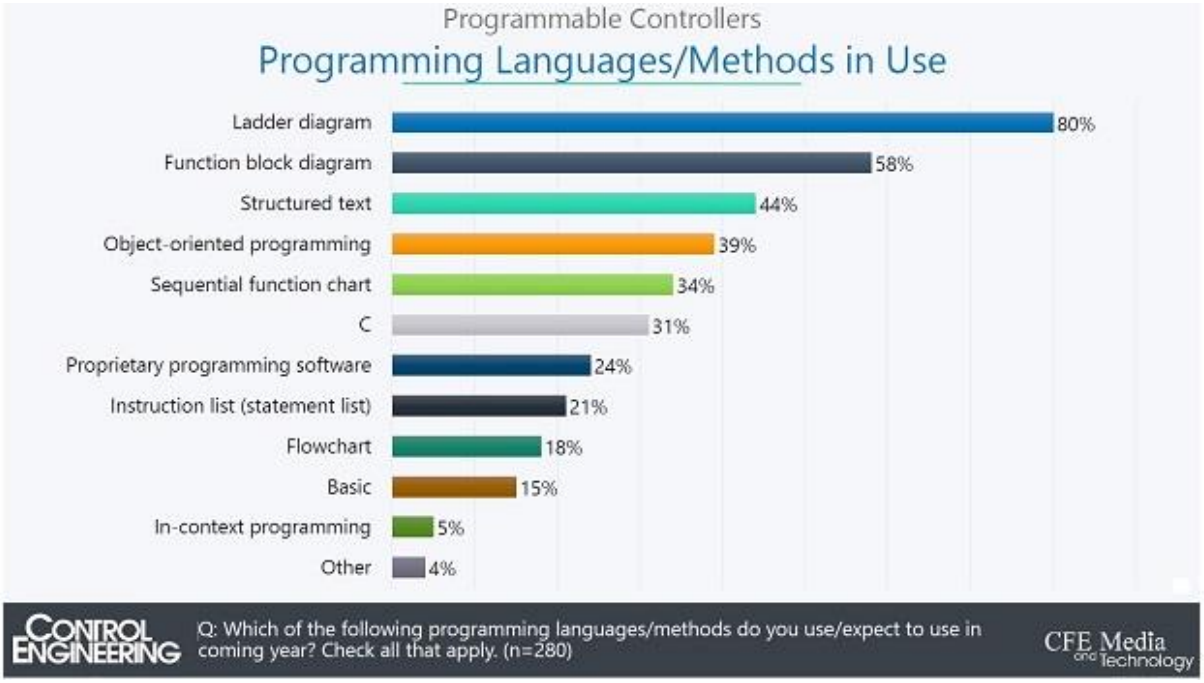

Рис. 3. Опрос журнала Control Engineering (2020)

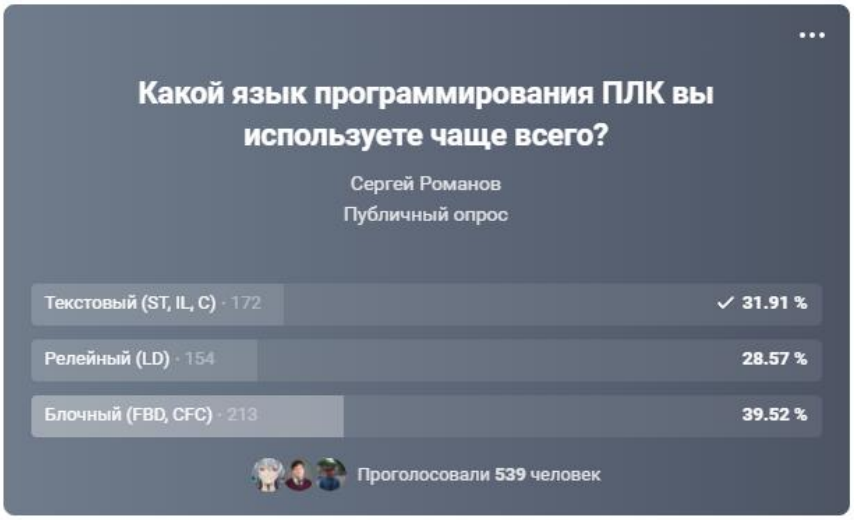

Рис. 4. Опрос Сергея Романова, проведенный им в рамках создания его книги (2019)

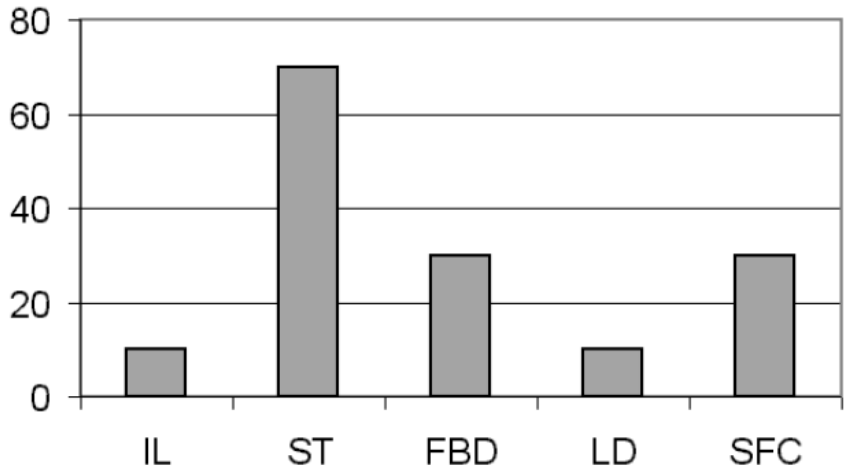

Рис. 5. Статистика популярности языков МЭК от разработчиков CODESYS на основании запросов в их техническую поддержку, опубликованная в 2006 в статье Игоря Петрова Отладка прикладных ПЛК программ в CoDeSys (часть 3)

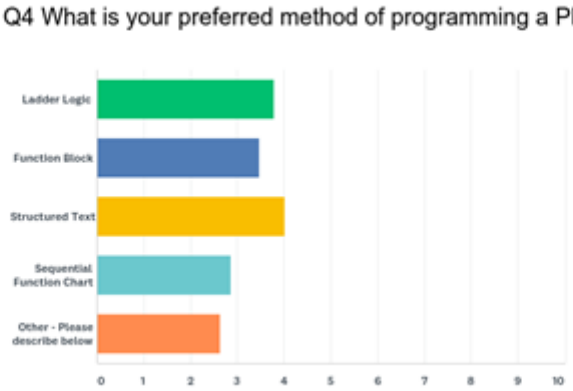

Рис. 6. Опрос организации PLCopen (2019)

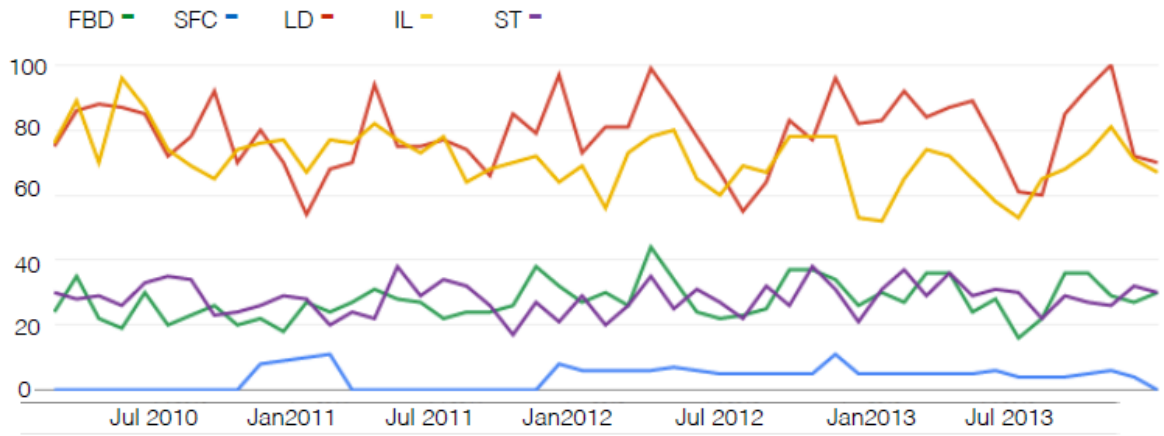

Рис. 7. Статистика относительной популярности языков МЭК в поисковых запросах Google, опубликованной статье Eduard Paul Enoiu (2010-2013)

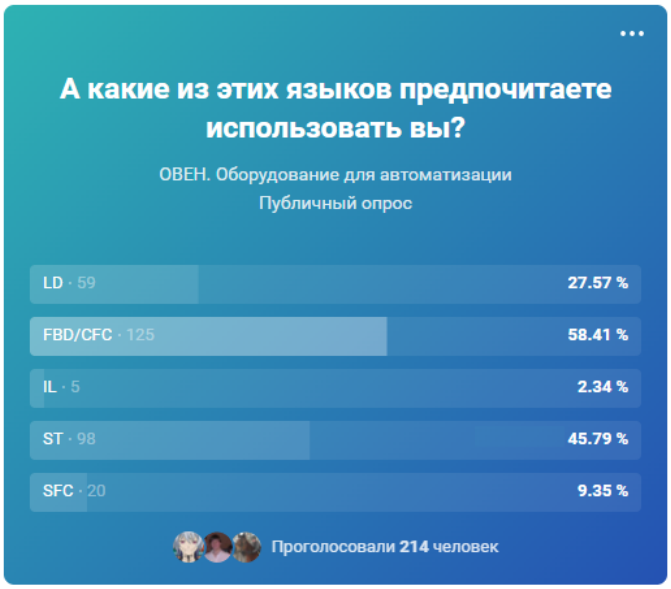

Рис. 8. Опрос в vk-группе ОВЕН (2022)

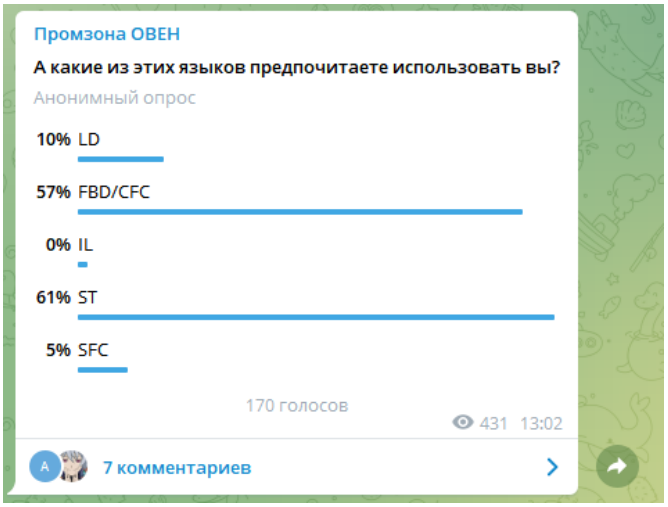

Рис. 8. Опрос в telegram-канале ОВЕН (2022)

Как можно увидеть – подобные опросы дают разные и зачастую противоречивые результаты. Вместо того, чтобы давать им какую-то оценку, мы решили пообщаться со специалистами, работа которых непосредственно связана с программированием контроллеров на языках стандарта МЭК 61131-3. Это люди, работающие в различных отраслях промышленности, решающие различные задачи и использующие для этого разные инструменты, имеющие разный опыт. Мы задали им один и тот же набор вопросов – и получили иногда похожие, а иногда совершенно противоположные ответы. Список вопросов, которые мы задавали, приведен ниже, а ответы на них от наших респондентов – в следующих пунктах.

Благодарим всех, кто уделил нам свое время, чтобы ответить на наши вопросы.

Список вопросов

1.Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования?

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответы респондентов

1. Сергей Романов

Сергей Романов – программист с 25-летним стажем, из которых 10 лет он работает в сфере автоматизированных систем управления. Автор книги Изучаем Structured Text МЭК 61131-3 и основатель yotube-канала с более чем 15 000 подписчиками. Основная используемая среда программирования – CODESYS.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: наверное, если поискать – можно найти настоящую причину. Я не знаю точно, какая идея была у разработчиков стандарта, но если предполагать – думаю, это сделано для того, чтобы дать разным специалистам возможность получить еще одну специальность – программист ПЛК. Для разработчиков схем релейной логики понятен LD, для архитекторов процессов понятен FBD.

Единственное – я бы выделил язык SFC. Если все остальные языки взаимозаменяемые (более или менее), то SFC это прямо реально отдельный от всех язык структурирования последовательных вызовов. Так что понятно, почему был создан SFC, так как это частая задача в АСУ.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: использовал SFC, CFC (*вариант FBD*), LD и ST. Последний из них предпочтительней. Он развязывает мне руки, делает программу читаемой, легче поддерживаемой, лучше организованной, легко копируемой. ST может всё, что и другие языки (даже SFC), так что используя его, можно даже не думать ни об одном другом языке.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: у ST есть недостаток – это отсутствие развития. Как мы знаем, все языки развиваются, выходят новые версии с новыми фичами. Но ST застыл во времени. Хотелось бы, чтобы он наследовал современные подходы. Например, короткий тернарный оператор ?:, возможность использовать значение строковых переменных для доступа к переменным по их именам. (*в стиле $a = "b", $b = 1, $c = $$a и $c будет 1, так как $aa – это обращение к $b*). И так далее. CODESYS не дождался, пока в стандарт ведут новые фичи, и ввел сам так называемый «расширенный ST». В нем есть UNION, __TRY__CATCH, __NEW, S=, R= и т.д. Хотелось бы, чтобы стандарт языка ST постоянно развивался.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: не сильно изменился, так как я и до этого много лет программировал. Но изменился в том направлении, что начал я с языка функциональных блоков, а теперь пишу только на ST.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: не приходилось. Мне всегда было интересно, как это работает. Я делал несколько попыток, но все были безуспешны – никогда не хватало вводных данных, как это делать. Но я вижу, почему я бы стал использовать С, если бы знал как. Думаю, по причине того, что в С я мог бы использовать разные готовые библиотеки. Например, я хочу подключиться к базе данных. На С есть уже клиенты, и пиши я на С – просто подключил бы библиотеку и всё (в прошивке ПЛК, соответственно, тоже нужна библиотека). Как правило, они платные, а написать такую же на ST – я даже не знаю, с чего начать. Если бы я знал, как писать на С, то я бы писал библиотеки для подключения к базам данных, реализации протокола MQTT и другое. Слышал, что еще Python сейчас потихоньку проникает на рынок – в связи с тем, что всё больше ПЛК работают на ядре Linux; думаю, на нем я бы писал по той же причине. Можно скачать библиотеки, поднять web-сервер, чуть ли не свой пользовательский интерфейс написать. Легко получить доступ к аппаратным ресурсам. Также на Python, думаю, люди будут писать, потому что знают его и не захотят разбираться с ограниченным по сравнению с ним ST.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: я лично считаю, что ООП вообще сильно переоценено, даже в прикладном программировании. Я начинал программировать процедурным подходом, и он решал всё. ООП – это не средство решения (так как все те же задачи можно решить и без ООП), а просто альтернативный подход к нему. Были у меня веб-проекты, в которых ООП было целесообразно и уместно, но в ПЛК, я считаю, процедурный способ рулит. Нет, конечно, наследование, методы – я это всё использую, но я не считаю, что это ООП. Частично использую возможности ООП для решения некоторых задач. Но мне кажется, это нельзя назвать использованием ООП для архитектуры программы. Это, скорее, локальные задачи. Это фичи, которые нужны, чтобы реализовать ООП, но сам ООП – это парадигма, когда ты представляешь какие-то сущности как объекты. Ну тут писать долго. Например, ты управляешь насосом. Ты создаешь объект насоса и если тебе надо его запустить – делаешь метод «старт», если остановить – «стоп». Но именно метод — ведь это действие, ну а если тебе нужно узнать его статус, то уже добавляешь свойство. Это ООП, но кто так делает в ST? Может, кто-то и делает – я его не осуждаю, но считаю это неразумным. Это не нужное усложнение архитектуры программы без явных преимуществ. ООП в ПЛК — это решение простых задач сложным путем.

Как правило, я применяю возможности ООП, когда мне нужно управлять несколькими одинаковыми устройствам. Например, несколько насосов или узлов соленоидных. И то, по большей части я, делаю структуры как отражение сущности. Но иногда и ФБ. Тогда некоторые действия я выношу в методы, но по сути – это не отражение реальных свойств и действий сущности, а просто распределение кода для более удобной организации, а методы, скорее, выполняют больше задачи процедур.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: да, сталкивался. И требования были именно по ST. По словам заказчика, он хотел, чтобы программа была именно на этом языке, потому что он каким-то образом узнал, что это самый лучший вариант.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: не думаю, что стандарт будет развиваться. Этот рынок не любит перемен. Думаю, и через 20 лет стандарт будет тем же. Но это, возможно, породит альтернативные решения. Ведь задачи всё современней – например, интеграция с облачными сервисами и IIoT – а стандарт к этому не готов. Нет, можно, конечно, и эти задачи решать, но это, на мой взгляд, скорее костыли, и то – отстающие на несколько лет. Мне нужно было MQTT в ПЛК намного раньше, чем первые библиотеки появились, и то – для тех ПЛК их до сих пор нет. А ведь это простая задача. Возможно, Python начнет вытеснять стандарт. Если смотреть в будущее, то первое, что приходит на ум – это искусственный интеллект (ИИ). Я думаю, что ИИ станет неотъемлемой частью любого ПЛК. В него будут встроены как процессоры для ИИ, так и библиотеки; можно будет настраивать его для выявления ошибок работы, нестандартных ситуаций и т.д. Также нас ждёт интеграция с новыми технологиями, которые сейчас активно развиваются в сфере домашней автоматизации – Zigbee, Z-Wave, голосовые ассистенты, мобильные решения.

2. Олег Евсеенко

Олег Евсеенко – ведущий инженер-программист с 5-летним стажем, разрабатывающий приложения для систем вендинга, растениеводства, вентиляции и ЖКХ. Его основные рабочие инструменты – CODESYS V3.5/2.3, OwenLogic, Visual Studio и Visual Studio Code.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: у каждого из этих языков есть свои преимущества и недостатки. Если брать, например, быстрый порог вхождения, то у ряда языков он выше. Зачастую программистами (то есть теми, кто пишет программу для ПЛК) могут быть люди без профильного образования, которым нужно перевести простенький алгоритм в код для выполнения на контроллере. Так, пройдя пару уроков, можно выучить тот же LD, знание которого может покрыть решение ряда задач.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: ST, FBD, LD. Основные ST и FBD. Без FBD не напишешь программу для ПР200 (хоть и анонсировалась поддержка ST). ST необходим для написания более «объемных» программ.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: прям у языков недостатки я бы не выделил; скорее, они есть у среды разработки.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: писать более универсальные программы, не под конкретную задачу, а как будто это платформа для решения набора однотипных задач, чтобы можно было функциональные блоки / функции использовать в других проектах. Оформление кода в виде библиотек для переиспользования.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: нет

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: нет, не приходилось. Не возникало потребности применения ООП в контексте программирования ПЛК. Сам по себе подход довольно мощный; всё зависит от задачи, которую нужно решать. Про возможности ООП в CODESYS с удовольствием бы лекцию послушал.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: да, настаивали на использовании FBD. Было и LD, но это было одна-две программы. Это были заказчики, которые хотели самостоятельно сопровождать этот код и графические языки должны были им помочь в настройке и изменении параметров. В случае с LD заказчику было удобнее работать именно с этим языком.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: неизвестно, будут ли ПЛК в том виде, в котором они существуют сейчас. Уже существуют попытки сделать тот же Arduino в промышленном исполнении. Так что, возможно, все будут писать на Python.

3. Вячеслав Лапшин

Вячеслав Лапшин – генеральный директор и главный специалист компании ООО «Быстрые проекты». Список компаний, в которых он работал, и реализованных им проектов описан в его блоге. Основные ПЛК, с которыми работает Вячеслав - Siemens, Beckhoff, Schneider Electric.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: основной целью стандарта является унификация. Руководитель на предприятии имеет меньше проблем с поиском лиц, которые умеют программировать, зная стандарт МЭК 61131-3. Таким образом, время на освоение чужого кода заметно снижается, и найти человека значительно проще. Так сложилось, что программисты ПЛК различаются по манере написания кода и типов использования языков МЭК 61131-3, что не мешает либо разобраться, либо переписать часть неработающего кода, если такое встретилось. Язык LAD, например, очень напоминает электрическую схему и хорошо подходит для тех, кто является электриком и совмещает работу программиста. Язык инструкций IL используется преимущественно при модернизациях, когда код наследуется со старого проекта, а само аппаратное решение меняется.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: приходилось использовать все существующие в стандарте языки. Если в процессе разработки присутствует заказчик, то для наглядности использую CFC (вариант FBD). Для более быстрой разработки программного продукта использую ST – там, на мой взгляд, большая гибкость

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: реализация языков в различных брендах – разная. Отличия особенно бросаются в глаза при отладке каких-то программ. Важно, чтобы значения переменных в коде были видны явно прямо там, где прописаны переменные. Если нужно значения смотреть в каком-то дополнительном окне, то скорость отладки сильно снижается, и это влияет на качество программного продукта.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: стараюсь использовать конечные автоматы, циклы – если использую заполнение каких-то массивов данных. В больших проектах стараюсь использовать производные типы данных – структуры и т.п.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: нет, не приходилось. Считаю, что данный язык нужен для реализации каких-то производных решений и функций вне деятельности промышленных ПЛК. Бывают задачи, которые не решить на базе языков стандарта МЭК 61131-3 и тогда решение делается на C#, но это уже несколько другой уровень – уровень HMI.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: по идее, ООП должно упростить и ускорить процесс разработки кода, однако только для тех, кто в совершенстве владеет данной идеологией. Тестировал, разбирался, но ввиду сложности у меня не пошло. Соответственно, разрабатываю код по старинке, как привык. Так получается быстрее. С радостью бы послушал какие-то видеоуроки по ООП.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: встречал. Аргументами была необходимость обслуживания и отсутствием компетенций по другим языкам. Обычно стараюсь идти на встречу.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: будет все немного быстрее и красивее отображаться. Появятся интерактивные подсказчики, которые будут предлагать варианты написания кода. Сам стандарт останется очень надолго.

4. Андрей Серых

Андрей Серых – главный инженер проектов, а также ведущий инженер в проектах. Опыт программирования – с 1994 года. Начинал с языков Visual Basic, TurboPascal, C, Delphi. Разрабатывал программы для диагностики компьютерного оборудования и драйверов. С 2000 года начал работать в средах Step 5 и Step 7, программируя ПЛК Siemens. В дальнейшем при знакомстве с различными линейками контроллеров работал в следующих средах разработки: MicroWIN, LOGO, RSLogix 5000, Codesys, PCS 7, B&R Automation Studio, TIA Portal, Unity PRO, GX Developer, GT Works и т.д. Имел опыт использования языков CFC (вариант FBD), S7-GRAPH (вариант SFC).

2. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: в своей практике сталкивался с работами коллег, встречал все варианты реализаций, для себя подчеркнул много плюсов и минусов в различных языках.

3. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: LAD, STL (IL).

4. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: есть как плюсы, так и минусы у каждого языка, описывать их не буду – слишком много есть чего сказать на данную тему, этого на отдельную статью хватит. Если одним предложением сказать: что нельзя сделать одним языком программирования, делается в другом.

5. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: на самом деле, там и меняться-то некуда. Библиотеки меняются от проекта к проекту и всё время улучшаются, стиль написания подстраивается под библиотеки. Изменилось только одно – со временем стал находить менее трудозатратные решения в написании кода и стал применять решения, которые используют меньше контроллерного времени на обработку задачи.

6. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: конечно, приходилось. Всегда зависело от задачи, которую необходимо было решить. Например: в среде разработки PCS 7 основной язык написания программы – CFC.

7. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: практически во всех задачах приходится использовать ООП. Иногда даже в самых простых задачах приходится использовать ООП.

8. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: сталкивались. Иногда требования прописаны в стандартах предприятия, иногда требования основывались на познаниях специалистов заказчика.

9. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: уже сейчас в TIA Portal происходят изменения – появляются библиотеки «от программиста». Нужно только добавить библиотечный блок и запараметрировать его, даже тюнинг предлагается. Вот, думаю, таким оно и будет – программирование будущего, но языки МЭК 61131-3 тоже останутся, для тех, кто не захочет пользоваться чужими библиотеками, или, например, платить за них.

5. Электрошаман

Электрошаман профессионально занимается разработкой и сборкой силовых щитов для квартир, дач, коттеджей и офисов. Подробнее об этом можно прочитать в его блоге. До работы с ПЛК программировал на ассемблере и С для микроконтроллеров. Использовал циклическую и таймерную модели (раз в пп времени что-то делать), простенькие конечные автоматы. Также программировал на Visual C++ для Windows и привык работать с текстовыми языками. До ПЛК программировал реле ABB CL (язык LD), Siemens Logo (язык CFC – вариант FBD). С ПЛК начал работать с 2016 года на CoDeSys 2.3, и с 2019 использует CODESYS V3.5 для программирования ОВЕН СПК1хх [M01]. Все проекты относятся к сфере автоматизации квартир и коттеджей и содержат простую дискретную автоматизацию: включение-выключения света, сбор информации от датчиков, дискретное регулирование с гистерезисом.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: не знаю, так как сам стандарт не изучал, а принял эти языки как данность. Моё личное мнение – что эти языки очень разные по стилю и структуре редакторов (как сделать редактор для графических схем и текста одновременно?) и можно было бы назвать МЭКовский стандарт как раз-таки «одним универсальным языком».

Мне нравятся разные языки МЭК, так как, по моему мнению, они удобны для специальных задач (язык SFC упрощает написание конечных автоматов состояний, например) или важны тем, кто переходит с других сред и способов разработки. Например, в старых релейных лифтах используются релейно-контактные схемы, и если такой лифт надо будет описать на ПЛК, то LAD отлично сгодится. Язык IL будет удобен тем, кто до этого программировал на ассемблере.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: так как до этого я работал с аналогом языка CFC, то начинал программировать ПЛК на нём, пытаясь в него вписать свою логику. Но так как логика в ПЛК была сложнее и требовала сложных и наглядных вычислений (чтобы можно было написать Y = (X / 20) + (11 * ValIO)), то я стал искать другие языки и пришёл к ST.

На данный момент основным языком у меня является ST, так как я привык программировать на текстовых языках в других средах разработки (микроконтроллеры, VC++, PHP): все текстовые языки похожи по своему стилю, и текст я всегда воспринимал лучше, чем графику. В текстовом языке можно писать подробные комментарии к каждой строчке кода, отделять что-то отступами, сворачивать (в редакторе) участки кода, чтобы они временно не показывались. Также тут можно описать более сложные алгоритмы проще и нагляднее, чем в графических языках.

На втором месте – CFC (*вариант FBD*), так как в нём есть удобная наглядность: можно на ST написать сложные FB (например, управление типовым вентилятором санузла), а потом нарисовать их в CFC и «подключить» только входные и выходные переменные, что и даст удобную наглядность.

Язык LAD я использовал только в реле ABB CL, и он мне оказался неудобен и менее нагляден, так как я привык мыслить функциями и блоками. Языки IL и SFC я не использовал.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: для меня недостатков нет. Были неудобные особенности среды разработки (такие, как автообъявление переменных, которое мне только мешает), но их можно отключить в настройках проекта.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: вначале я работал с CFC и пытался делать на нём то же самое, что делал в Siemens Logo – рисовать одну большую «схему» на всю страницу, используя простые вспомогательные FB для простых функций. Это было неудобно и ненаглядно, и постепенно я пришёл к продвинутым FB, которые, например, управляют не одной группой света, а сразу всей комнатой из 10 групп света максимум (неиспользуемые входы-выходы мы оставляем пустыми, незадействованными).

Все FB я сначала тоже писал на CFC, потому что не до конца понимал то, как в ST вызывать другие FB в виде текста (мне было привычно рисовать «квадратик» и подавать «сигналы» на входы и выходы). После того, как я оценил удобства текстовой записи вызовов FB в ST и то, как это работает – я начал писать почти полностью на ST, а на CFC рисовать только те части программы, которые будут простыми и наглядными.

Сам стиль написания программ у меня сложился из других языков и сред разработки – понятно называть переменные и комментировать код, группировать разные объекты в дереве ресурсов по папкам и подпапкам. Эту часть стиля я перенёс на CoDeSys. Также я пытался в ST применить методы, характерные для уже известных мне сред – поэтому возникали вопросы про работу с памятью, объявление и инициализацию структур и FB, и т.д.

Новым стало то, что я начал делать много удобных FB под свои задачи, и стремиться к их универсальности: например, создал сложный FB для работы с одним каналом модуля аналоговых входов (ОВЕН МВ110-8А), который сразу проверяет статус связи с модулем, статус опроса датчика, допустимость его значений, считает число опросов, число ошибок и формирует понятную строку для отображения значения на экране (например, «32,87 °C» или «Err.» в случае ошибки) и понятную строку статуса измерения канала («Ошибок нет», «Обрыв датчика» и т.д.).

После применения таких FB код становится похож на большую копипасту однотипных вызовов FB для обработки каналов.

Ещё я научился мыслить очередями и событиями ПЛК, когда работал с менеджером тревог и отправкой сообщений (SMS, E-Mail): оказалось удобным создать очередь на отправку и приём SMS-ок, в которой по таймеру (чтобы модем успел принять-передать данные) выбираются новые сообщения и ведётся их обработка.

Следующим шагом было (и идёт) стремление к автоматизации вызова однотипных FB подряд при помощи создания массива FBшек и прохода по его вызовам в цикле – например, для опроса нескольких устройств Modbus через библиотеку OwenCommunication. На эту идею меня навело

программирование в VC++, где можно было создавать массивы или коллекции указателей и объектов.

Есть будущие мысли сделать автоматическую генерацию типового кода опроса устройств и разбора их битовых масок при помощи самописной сторонней программы, но я пока не уверен, что смогу написать эту программу быстрее, чем сделать всё руками.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: именно для программирования ПЛК – нет, так как я сразу стал работать с CoDeSys и ПЛК ОВЕН.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: я не уверен, что мне хочется видеть ООП в программах для ПЛК (но я сужу по своей сфере, где большинство задач по автоматизации решается текущими языками и при помощи FB). Всё-таки ПЛК выполняет программу в цикле, и когда про это знаешь – то хорошо понимаешь последовательную структуру языков и вызова FBшек.

ИМХО, ООП в плане наследования FBшки от другой FBшки только всё запутает (для меня). На других языках я несколько раз наступал на грабли, связанные с тем, что какой-то объект (класс) выполняет стандартную задачу, и его можно только наследовать. А если нужно изменить то, базовое, стандартное поведение – то этого сделать не получится без изменения базового класса. Поэтому если нужен один класс с оригинальным поведением, а другой – доработанный – то приходится копировать этот класс и изменять его так же, как если бы это был обычный FB. Я не представляю, что можно было бы наследовать в ПЛК. Класс котельной? Класс вентиляционного шкафа? Класс ПИД-регулятора? А для чего?

При этом я приветствую использование ООП внутри CoDeSys для описания объектов устройств, библиотек и прочих внутренних объектов ядра. Такие объекты поставляются вместе с ядром и тут нет смысла ругаться на то, что их наследование мешает изменить базовое поведение этих объектов.

Хочется рассказать историю, с которой я столкнулся, работая с некоторыми заказчиками. Программирование под Arduino и подобные им среды предполагает позднее связывание IO-каналов и объектов прямо в коде программы, примерно так:

```
VAR
myDimmer : CDimmer;          // Объявили переменную объекта
END_VAR

myDimmer.BindButton(DI4);    // Назначили вход кнопки
myDimmer.BindLamp(AO10);     // Назначили выход на лампу
```

Если такие люди переходят на ПЛК, то они как раз-таки и спрашивают о том, почему в ПЛК так нельзя и где здесь ООП и создание экземпляров объектов.

Мне удобен стандартный подход FB, где можно не просто привязать одну кнопку ко входу «диммера» из моего примера, а вызывать FB диммера с нужными значениями – например, «подав» на его кнопку несколько сигналов:

myDimmer (Button := DI4 OR UIButton, Lamp => AO10);

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: нет, не встречался. Часть моих заказчиков разрабатывали программы сами без моего участия, а в остальных случаях я пишу программы сам на ST и CFC.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: я не могу предположить. Возможно, в ПЛК придёт концепт программирования умных домов в виде событийных связей: когда вместо переменных есть некие объекты (возможно, тут и сгодится ООП) определённых типов: «Кнопка», «Выключатель», «Нагрузка», «Датчик температуры» – которые можно разместить на экране и связать их между собой линиями.

По этим связям система будет понимать то, что от неё хотят. Например, если вы связали «Кнопку» и «Лампу» – то лампа должна переключать своё состояние по кнопке, а если сюда же параллельно «Кнопке» дорисовали «Выключатель», то система поймёт, что одна и та же «Лампа» может управляться как по кнопке, так и по выключателю (по фронтам/спадам).

Если такое появится – то я бы хотел, чтобы это было ещё одним языком МЭКовского стандарта, а не новым языком, который заменит все остальные.

6. Владимир Дубилей

Владимир Дубилей – инженер-программист с 5-летним стажем. Основная среда программирования – CODESYS 2.3 и 3.5, основной язык программирования ST. Область промышленности — пищевая промышленность.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: проекты по автоматизации исполняются работниками, имеющими различную специализацию. Программистам на языках высокого уровня ближе ST; те, кто имел опыт с релейной логикой, выберут LD и т.д.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: приходилось использовать ST, CFC (*вариант FBD*) и FBD. Мой основной язык ST, ранее программировал на C++.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: нет, к синтаксису языка ST претензий нет. Ничего бы не менял.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: при переходе на CODESYS V3.5 стало возможным использование ООП – это самое замечательное изменение в стиле написания программ на МЭК 61131-3.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: нет, не приходилось.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: с некоторого времени использую постоянно. ООП значительно упрощают написание проектов, отладку кода. Расстраивает отсутствие обучающих материалов по правильному и безопасному использованию ООП при написании проектов по автоматизации.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: нет, не сталкивался.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: на мой взгляд, основное развитие получит язык ST; будет развиваться его объектно-ориентированная часть. В среде разработки появится возможность создания шаблона программы — возможность сгенерировать машину состояний, набросать основные исполнительные механизмы проекта, автоматизированное подключение модулей ввода-вывода и т.д.

7. Сергей Шугаев

Сергей Шугаев - руководитель компании ПРОЕКТ-П с более чем 10-летним стажем. В работе использует ПЛК разных производителей, в основном ОВЕН, Siemens, Wago. Программы разрабатывает в Codesys и TIA Portal.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: в определенных задачах тот или иной язык дает свои преимущества по скорости разработки и наглядному восприятию кода. В одной задаче удобно использовать язык релейных диаграмм, в другой структурный текст, в третьей комбинацию языков.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: ST, LD, CFC (*вариант FBD*). Основной язык – ST. Язык более гибкий по сравнению с другими языками; сложные алгоритмы гораздо проще реализовать на нем, чем на других языках. Дополнительные языки – LD, CFC. С помощью данных языков можно быстро и наглядно разработать простые алгоритмы управления.

Самый лучший вариант, на мой взгляд – декомпозировать задачу на мелкие подзадачи и использовать для реализации комбинацию языков. Простые подзадачи делать на языках LD и CFC. Более сложные реализовывать на ST.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: сама структура языков меня устраивает, но в разных средах программирования существуют свои фишки. Например, для объявления ФБ R_TRIG (F_TRIG) в Codesys необходимо добавить соответствующий функциональный блок в тело программы. В других средах программирования достаточно поставить «стрелку вверх» или «стрелку вниз» на соответствующем контакте языка LD без объявления ФБ, и это быстрее и удобней.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: со временем я перешел на венгерскую нотацию переменных, практически отказался от сокращений имен переменных, больше стал уделять внимания структуре и меньше стал комментировать код.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: в далеком прошлом пробовал писать программы на C++ для ПЛК ICP-CON – сложно и неудобно отлаживать. Был опыт программирования GSM-терминалов на языке Java – интересно, много ООП. Также программировал контроллеры МЗТА и Segnetics. Данные ПЛК имеют свою среду программирования и графические языки, похожие на МЭК, специфические и крайне неудобные.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: никогда не использовал принципы ООП при программировании ПЛК. Как правило, каждая задача автоматизации уникальная и сводится к фиксированному количеству входов и выходов, которые, в свою очередь, так или иначе приходится привязывать «ручками», поэтому не вижу возможностей применения ООП для программирования ПЛК. Быстрее написать обычный процедурный код, чем придумывать абстракции на уровне ООП. ПЛК – не компьютер, и это надо понимать.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: не встречался.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: я бы хотел видеть в первую очередь развитие сред программирования (IDE). Если сравнить любую существующую IDE для программирования ПЛК с IDE IntelliJ IDEA, то будет сразу понятно, в каком направлении должны развиваться IDE для ПЛК. Через 20 лет восстанут машины из пепла ядерного огня и пойдет война на уничтожение человечества… (шутка)

8. Андрей Рыбалко

Андрей Рыбалко – инженер-программист, руководитель отдела АСУТП. Занимается разработкой систем управления и визуализации технологических процессов – в частности, построением систем стендового оборудования для испытания авиационных агрегатов, разработкой пультов управления агрегатами и механизмами. В основном использует среды разработки CoDeSys V2.3 и CODESYS V3.5, которые поддерживают достаточно широкий круг ПЛК.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ:

Не вдаваясь в теорию и историю происхождения стандарта МЭК 61131-3 скажу от себя, что если за более чем 20 лет этот стандарт выжил, и все пять языков имеют практическое применение – значит, оно кому-то нужно.

К примеру, на производстве имеются автоматические конвейерные линии с кучей последовательных операций, протяженностью в десятки и сотни метров. Эти линии нафаршированы датчиками, исполнительными механизмами, приводами и т.д. Возникает аварийная ситуация, участок линии останавливается, и, соответственно, весь технологический процесс стопорится. И вот технический специалист обслуживающего персонала подключается к линии, видит структуру файлов с кодом на ST, и, лихорадочно пытаясь отыскать неисправность, в душе благодарит разработчика. Такой код затруднит поиск неисправности; в то же время код, написанный на LAD и FBD, в кратчайший срок помог выявить причину, и сэкономить владельцу предприятия деньги, а специалисту –репутацию :)))

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: лично я использую ST, так как он обладает многими признаками высокоуровневого языка, достаточно гибкий, и на нем можно реализовать все, что нереально реализовать на графических языках (FBD/LAD/SFC)

На практике приходилось использовать LAD на производстве с множеством конвейеров; SFC вставляем там, где есть определенный алгоритм последовательных действий (включений/выключений механизмов)

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: в целом, языки стандарта МЭК 61131-3 считаю достаточно надежными и практичными. Скорее, не из-за их недостатков, а из-за специфики программы этого стандарта обладают плохой переносимостью, кросс-брендовость практически не имеет смысла. В стандарт МЭК 611313 добавил бы C/C++ для удобства.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: код новичка от кода профи легко отличить. Как ни странно, зачастую у профи всё более упорядочено, понятно, присутствуют комментарии, имеется культура программирования. Со временем приходит понимание того, что код помимо надежности должен обладать читаемостью, легко восприниматься, снижать нагрузку на зрение и т.д. Я полагаю, что кроме решений, направленных на обеспечение безопасности и эффективности ПО, качество ПО можно повысить, учитывая и соблюдая :

- общие правила при разработке ПО;
- правила компании, сотрудниками, которой являются участники рабочей группы;
- правила, созданные самими участниками рабочей группы.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: (*респондент не предоставил ответа на этот вопрос*)

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: решаемые мной задачи малой и средней автоматизации на данный момент не требуют применения ООП. В то же время является большим плюсом возможность создавать классы, объекты и пользоваться преимуществами ООП. Этот метод оправдает себя при большом количестве оборудования, которое можно будет организовать как объекты и «позаворачивать» в классы.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: лично у меня нет необходимости привязываться к определенному языку. Решение принимается ситуативно.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: думаю, всё так и останется. Критические места программ будут реализованы на низкоуровневых языках. И, возможно, ПЛК будут включать в себя как минимум 2 процессора – на одном будет реализована логика, а на втором – задачи пользовательского интерфейса.

9. Алексей Осинский

Алексей Осинский – программист\техлид. Опыт программирования ПЛК ~7 лет (CODESYS и чисто ознакомительный обзор других МЭК сред разработки).

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: чтобы снизить уровень входа для инженеров. Раньше, когда основная сложность системы была в «железе», было проще научить инженера программировать, чем программиста разбираться в тонкостях техпроцесса.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: CFC (*вариант FBD*)\ST\SFC (облегченный)

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: каждый из них хорош для решения своих задач. Хорошо, когда они все поддерживаются средой разработки и доступны одновременно в одном проекте.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: сначала было слово. Точнее картинка. CFC. Что странно, ведь до того, как я начал разрабатывать ПО для ПЛК, у меня был опыт разработки ПО на Delphi, поэтому начать на ST должно было быть проще.

Но пока «проекты» (а точнее решаемые задачи) были маленькие – было что-то увлекательное в протягивании связей между компонентами. Потом, когда проекты стали сложнее (а точнее перечень решаемых задач сместился от автоматизации техпроцессов к более системным вещам), добавился ST. Позже был проект, в котором требовалась машина состояний. Я написал ее на ST, и спустя какое-то время после сдачи проекта (не помню, по какой причине) клацал мышкой в CODESYS и наклацал её же на SFC. Был озадачен и много думал, почему же я никогда не пробовал использовать этот дивный и такой понятный язык?

Потом был небольшой период, когда я пытался найти применение SFC прям везде: ведь он же такой выразительный!

С тех пор обычный проект включал все 3 языка:

- на верхнем уровне CFC, чтобы было наглядно видно связи между компонентами;
- основная логика на ST, потому что быстро и читабельно;
- машины состояний на SFC.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: нет

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ:
Приходилось ли вам
Приходилось.

Что вы думаете о применении ООП в контексте программирования ПЛК?
Это божественно.

если да – то для каких задач?

Я даже начинал писать статью на эту тему. Но дошло до написания примеров, и я сдулся. После нормальных человеческих сред разработки возвращаться в CODESYS прям совсем не хочется.

Если кратко: архиватор и другие системные проекты. Но это не значит, что ООП нет применения и в автоматизации техпроцессов. «Двигатель», «Автоклав» и прочие сущности – это ведь явные объекты.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: нет

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: затрудняюсь ответить, не слежу за развитием направления. Но, честно говоря, мне бы хотелось, чтобы комфорт работы в IDE довели до уровня сопоставимого с IDE здорового человека, и я был бы счастлив.

10. Владислав Зинько

Владислав Зинько – ведущий инженер-программист бюро систем автоматизации. Профессиональный опыт – 5 лет. Общий – около 8 лет. Основная рабочая среда разработки: CODESSYS V3.5. Работал также в Unity Pro (с v3 до v11) и TIA Portal. 4 года специализировался на программном обеспечении автоматизации производственных процессов разной величины.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: прежде всего, эти языки призваны упростить вхождение в процесс программирования ПЛК «непрограммистами». Те, кто умеет читать и разрабатывать релейно-контактные схемы, сможет достаточно просто написать первую программу на языке LD (LAD). Далее перейти на FBD и так далее. Кто знаком с ассемблером – быстро приспособится к IL. Кроме того, это дает выбор пользователю. Для каждой задачи, по сути, есть наиболее подходящий язык. Для взаимоблокировок, прописанных только булевыми переменными – LD (как наиболее наглядный) и т.д.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: все, помимо IL. С этим языком сталкиваться не было причин. Основным языком для себя определяю ST. Именно на нем с наименьшими трудозатратами можно решить любую задачу – язык максимально гибкий и полный. Это не отменяет того факта, что в проектах помимо ST использую и SFC, в меньшей мере – все остальные языки.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: скажу за свой основной язык (ST): мне не хватает динамической работы с памятью. Учитывая возможности, которые дает CODESYS V3.5 – на мой взгляд, это было бы неплохим подспорьем. Но я прекрасно отдаю себе отчет, что навряд мы получим когда-либо готовый встроенный функционал для этого. АСУ ТП не та область, где простительны игры с памятью устройств. Но как возможность – было бы неплохо иметь.

Второй момент – переопределение функций. Во многих проектах была бы незаменимая вещь. Есть обходной путь, конечно – использование переменных __System.AnyType. Но это выглядит уже как «костыль».

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: стал больше закладывать универсальности и возможности масштабирования почти в каждый юнит программы. Проекты теперь разбиты на множество максимально самостоятельных модулей, которые с легкостью «переезжают» в другой проект. Код на 99% написан на ST.

Стал больше внимания уделять также удобству пользовательского интерфейса, если речь идет о проектах с визуализацией.

Венгерская запись в именовании переменных доведена до автоматизма.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: приходилось использовать инструменты веб-разработки для поднятия веб-сервера на базе ПЛК Simens. В этом ПЛК сервер не работал автоматически, а требовал для себя файлы веб-страниц, инструкций для серверного обмена дынными и т.д.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: да, приходилось и приходится. Для реальных объектов – наследование функционала механизмов. Если придерживаться концепции универсальности, переносимости и т.п., каждый механизм был наследником базового механизма с базовым интерфейсом управления. Соответственно, основной алгоритм, в большинстве случаев, «не знал» чем именно управляет, т.к. получал массив базовых интерфейсов. Это очень выгодно для случаев конфигурирования тех же самых выходов ПЛК с визуализации. Сгорел выход – перекинули механизм на свободный выход, переназначили и все продолжает работать.

В pet-проектах тоже часто использую наследование и полиморфизм. Это, к слову, очень удобно при разработке ПО для тестирования ПО. Уменьшает количество копируемого кода и экономит время. В целом – я за ООП. Но здесь как с религией – не каждому по душе то, что по душе тебе.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: да. В нескольких крупных организациях сталкивался с тем, что язык программирования выбирал заказчик. Связано это было с тем, что все исходники передавались им и дальше их сопровождали уже их программисты, которые, видимо, были компетентны в своей конкретной области МЭК 61131-3.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: думаю, через 20 лет программировать ПЛК уже будут не на МЭК-овских языках. Скорее на Python или что-то около того. Сейчас уже наблюдается некий кризис кадров в нашей области. Но если перевести ПЛК на программирование языками IT-сектора, то найти сотрудника не составит большой проблемы. Это также повлияет на расширение комьюнити программистов АСУ ТП.

Так или иначе, МЭК 61131-3 уйдет в историю, на мой взгляд. Рано или поздно, но он сменится на стандарт, в котором будут описаны не менее «строгие» языки.

11. Алексей Муравицкий

Алексей Муравицкий – ведущий специалист компании ОКБ АМУР 3. Выпускник Первого инженерного факультета Омского Танкового Инженерного Института (курс майора Свечкова 2003 г). Окончил курсы по программированию ОВЕН ПЛК1хх и ОВЕН СПК. Программирует в средах CoDeSys V2.3 и CODESYS V3.5 уже 7 лет. Реализовал более 100 проектов на ПЛК и ПР (в основном, на ОВЕН СПК и ПЛК210) – всевозможные насосные станции и установки, тепловые пункты, координатные станки, сложные технологические установки и т.д, и т.п.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: у каждого языка есть своя манера алгоритмирования процесса, и для каждого программиста есть тот или иной удобный способ сделать объект на любом языке.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: CFC (*вариант FBD*), ST. CFC – как правило используется для общей сборки в PLC_PRG. Все остальные составные части проекта делаются объектно-ориентированным методом на языке ST. Встречал людей, которые пишут на IL и на языке релейных схем (очень много), тоже хочу освоить

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: каждый язык имеет свои плюсы и минусы. Использование сильных сторон языков позволяет в одном проекте писать на нескольких языках (например, работать с массивом удобнее в ST, а наглядно собрать конечные блоки объектов на CFC), а мой коллега до кучи пишет конечный автомат на LD.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: двигаюсь больше в сторону объектного программирования.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: да – язык Дракон. Он позволяет алгоритмировать любой процесс в кратчайшие сроки. Для примера: на один хорошо отработанный функциональный блок объекта с алгоритмом средней сложности и объемом в 120-170 строк программного кода и 200 строк комментариев на языке ST уходит 2 дня, на языке CFC – 3 дня, на языке Дракон – 4-6 часов.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: да, объектный подход — это основной подход в работе. В CODESYS он отличается от подобного подхода в программировании (например, в той же VS 2020). Этот момент создает для меня сложности в понимании и использовании, например, свойств и методов.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: нет, заказчик ни разу не настаивал. Более того, многие заказчики с интересом относятся к нашей среде программирования (Дракон), так как она позволяет формировать графический документ, который понятен практически всем. Часто заказчик запрашивает алгоритм в графическом виде из среды Дракон как отчетный документ.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ:

- использование трибитной системы вместо двоичной;
- смещение полностью в зону ООП;
- частичное использование технологий искусственного интеллекта (возможно, придется скоро обучать ФБ, как себя вести на объекте, а не писать алгоритм);
- ПЛК получат возможность обрабатывать изображение (техническое зрение — это решит много технологических проблем, связанных с более глубокой автоматизацией процессов производств)

12. Роман Радченко

Роман Радченко – инженер-программист с опытом работы в 2 года. Основные используемые среды разработки – CODESYS V2.3/V3.5, OwenLogic, TRACE MODE, WinCC.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: **IL** – атавизм 1993 года, остался, потому что удобен программистам старой (очень) закалки. **FBD** – удобен для понимания программирования людям, далёким от написания кода в его стандартном представлении. **SFC** – даёт возможность без усилий переносить блок-схемы в код проекта, так же удобен для реализации простых машин состояний. **ST** – основной (по моему скромному мнению) язык программирования, т.к. он читаемый, гибкий, простой в отладке и имеет самые обширные возможности. **LD** – нужен для конвертации в код трижды пересканированных начерченных от руки релейных схем из 20-го века.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: в реальных проектах использовал 3 языка: ST, SFC + ST, CFC (*вариант FBD*).

ST – наиболее удобный. В большинстве случаев написание одного и того же алгоритма займёт на ST намного меньше времени, чем на любом другом языке; отладка намного быстрее, чем на любом другом языке. Циклы, кейсы и прочая радость, недоступная тому же FBD.

SFC + ST – по началу SFC кажется удобным заменителем кейса с таймерами и входными/выходными триггерами для выполнения единичного действия, но в процессе работы мне пришлось отказаться от него в пользу чистого ST, так как быстро реализовать очередную «хотелку», не уперевшись в негибкость SFC, проще и быстрее на чистом ST.

CFC – также по началу казался удобным языком для чего-то простого, с минимальной логикой. Но как показывает практика – быстрые правки на объекте превращают «код» на CFC в нечитаемый кратно быстрее, чем на том же ST.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: из больших недостатков отсутствие динамических массивов (опираюсь на опыт в CODESYS).

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: полный уход от всех языков, кроме ST. Более высокая степень декомпозиции кода, и максимальный отказ от VAR_INPUT/OUTPUT/IN_OUT, если есть такая возможность.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: использовались только скрипты с командами Linux. К примеру: одной собранной строкой намного удобнее копировать файлы, чем библиотечными блоками и др.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: необходимо, использую повсеместно. ООП значительно сокращает время на переработку кода и плодит независимые программные компоненты, которые намного проще импортировать в другой проект.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: нет, такого не встречал

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: по моему мнению, стандарт МЭК 61131-3 будет и дальше использоваться для программирования устройств в АСУТП, так как набор языков достаточной обширный и имеющийся функционал покрывает большинство потребностей.

13. Александр Головащенко

Александр Головащенко – ведущий инженер-программист АСУ ТП с опытом работы в 8 лет. Основная среда программирования – CoDeSys V2.3.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: если кратко: так исторически сложилось. На момент создания стандарта МЭК разработчики стандарта столкнулись с большим количеством похожих языков программирования, которые использовали разные производители. И для обеспечения универсальности и совместимости стандарта с текущей ситуацией на рынке им пришлось включить в него 5 наиболее распространенных на то время языков.

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: если нет других требований – я использую только язык ST. Данный язык наиболее функционален и гибок относительно остальных языков стандарта. А совмещение в проекте нескольких языков считаю излишним усложнением структуры проекта. Гораздо проще работать, когда все написано в одном стиле с использованием одних и тех же инструментов.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: основной недостаток языка ST – это отсутствие ООП и отсутствие динамических массивов. Другие языки стандарта МЭК (FBD, LD, SFC, IL) я не считаю полноценными языками программирования из-за отсутствия возможности работы с массивами и структурами данных, а также отсутствия циклов.

Но большинство недостатков относится не к самим языкам, а к средам программирования. Слабые по современным меркам библиотеки и слабая интеграция с другими современными инструментами разработки ПО тормозят процесс разработки и поддержки продукта.

Вот часть функционала, который присутствует в любом другом современном языке программирования, но не поддерживается (или поддерживается слабо) в средах программирования ПЛК:

- ООП;
- динамическое выделение памяти для массивов;
- работа со строками и текстами, регулярными выражениями, юникодом;
- поддержка мультиязычности интерфейсов;
- интеграция юнит-тестов, измерение тестового покрытия, измерение времени выполнения кода;
- работа с популярными форматами данных (JSON, XML и т.п.);
- поддержка распространенных решений для IOT (например, протоколов HTTP, HTTPS, MQTT, gRPC);
- работа с базами данных, как локальными, так и удаленными;
- интеграция инструментов контроля версий.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: с течением времени я стал все больше создавать более высокоуровневые конструкции в проекте. Это значительно ускоряет и упрощает процесс разработки, а также поиск и исправление ошибок.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: я использую язык Python для расширения функционала среды разработки и автоматизации однотипных действий.

Например, я использую скрипты, которые генерируют однотипные части кода ST из файла с набором параметров для данного кода (например, для подключения входов / выходов, датчиков, модулей расширения, обработки регистров Modbus, обработки параметров HMI и других однотипных действий в зависимости от требований проекта).

Также я использую скрипты для подготовки и выполнения юнит-тестов и дальнейшего сбора результатов тестирования.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: да, приходилось.

ООП является удобным инструментом абстракции. Наиболее важные аспекты ООП для программирования ПЛК, на мой взгляд – инкапсуляция и полиморфизм.

Инкапсуляция позволяет работать на более высоком уровне абстракции, а также способствует повторному использованию кода. Полиморфизм позволяет выполнять однотипные действия для всех объектов одинаково, не заботясь о том, к какому классу они относятся (так, например, массив различных датчиков можно обрабатывать одинаково, не думая о том, какого именно типа каждый датчик). Оба этих принципа значительно уменьшают объем кода и количество нюансов, о которых нужно помнить в процессе разработки.

Такие сущности как «функциональные блоки» в CoDeSys позволяют инкапсулировать в себе данные и поведение объектов аналогично классам в ООП, но при этом реализация ФБ более громоздка и запутана, т.к. ФБ не имеют разделения на методы, а вся логика пишется одним куском кода. Аналогичная реализация с использованием классов ООП была бы более простой для понимания и поддержки. Также у ФБ отсутствует поддержка полиморфизма, что не позволяет одинаково обрабатывать схожие ФБ.

Случаи, для которых применял ООП:

- обработка сигналов с датчиков разных типов (термосопротивление, NTC, PTC, токовые 4...20 мА, напряжения 0…10 В, датчики с цифровым интерфейсом, импульсные датчики и т.п.);

- реализация протоколов обмена данными (протокол ОВЕН, Modbus RTU, Modbus TCP, HTTP и т.п.);
- управление исполнительными механизмами (нагревателями, охладителями, электроприводами, вентиляцией);
- логирование данных;
- обработка аварийных ситуаций;
- юнит-тесты.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: иногда сталкиваюсь с ситуацией, когда Заказчик хочет обеспечить себе возможность самостоятельно поддерживать и дорабатывать проект, и сотрудники Заказчика имеют квалификацию только в каком-то определенном языке МЭК. Также встречаются ситуации, когда у Заказчика есть внутренние стандарты предприятия, которые определяют язык программирования.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: на мой взгляд, необходимо внедрять инструменты, которые повышают уровень абстракции кода – например, ООП. Также необходимо развивать и расширять количество стандартных библиотек, которые поставляться с средами разработки ПО.

Например, язык Python сейчас является столь популярным по нескольким причинам:

- во-первых, он высокоуровневый и имеет дружественный лаконичный синтаксис. Это сокращает число строк кода, необходимых для достижения результата;
- во-вторых, он «с батарейками в комплекте», т.е. Python поставляется сразу с широким набором стандартных библиотек, которые уже реализуют большое количество распространенных задач.

Подавляющее большинство современных языков программирования включают в себя концепции ООП, высокоуровневость и стремятся к упрощению синтаксиса. Эти факторы ускоряют разработку и поддержку программного обеспечения и, следовательно, сокращают затраты и делают продукт более конкурентоспособным на рынке.

14. Борис Калинин

Борис Калинин – ведущий инженер АСУ ТП с более чем 10-летним стажем. Работает над разными проектами на базе ПЛК. Основное направление — теплоэнегретика. Борис автоматизировал около 500 тепловых пунктов и котельных разных масштабов и полностью автоматизировал тепловые сети Волгограда. Кроме того, автоматизировал несколько десятков больших частных домов, в основном – тепло- и водоснабжение на базе тепловых насосов собственной разработки Ovanter. Также разрабатывает автоматику для серийно выпускаемых фандоматов (аппаратов для сбора бутылок и банок) компании «Группа Тарас». Реализовал много проектов по мониторингу и автоматизации зданий, водоснабжению, вентиляции, электроснабжению, систем для буровых установок нефтяных скважин, заводских цехов, станков, классификации транспорта.

Основные среды программирования для Бориса – CODESYS V2.3 и V3.5. Используемые ПЛК – в основном, ОВЕН, иногда Berghof, Beckhoff, Schneider Electric, ABB. Также он программирует ПЛК Phoenix Contact в PC WORX.

Больше информации о Борисе и реализованных им проектах можно найти на его сайте и старом сайте.

1. Почему, на ваш взгляд, стандарт МЭК 61131-3 описывает 5 различных языков программирования, а не один универсальный?

Ответ: вроде так исторически сложилось, как и всё в этой консервативной сфере. Изначально для более легкого вхождения для электриков (LD) и технологов (FBD, CFC).

2. Какие языки стандарта МЭК 61131-3 вам приходилось использовать в реальных проектах? Какие из них являются для вас основными и по каким причинам?

Ответ: сталкивался со всеми. Работаю на ST и МЭК-SFC, изредка CFC. Основа — ST, он лаконичен и что-то серьёзное и читаемое на LD, FBD, IL не сделать. Изначально в университете программировал на C# и микроконтроллеры на C++, поэтому проще было перейти на ST.

Язык SFC использую для фандоматов и заводского оборудования, где есть выраженная последовательность действий.

3. Есть ли, на ваш взгляд, у используемых вами языков недостатки? Чтобы вы хотели изменить в них?

Ответ: для моих задач возможностей хватает и не знаю, что изменять. С другими языками почти перестал работать, поэтому не могу сравнить.

4. Как изменился ваш стиль написания программ на языках МЭК 61131-3 с течением времени и приобретением опыта?

Ответ: стал всё больше использовать объектный подход и сильнее разбивать код на функциональные и логические блоки, сильнее структурировать и оптимизировать его. Всё больше применяю язык SFC из-за его особенностей и удобства в некоторых случаях.

5. Приходилось ли вам использовать для программирования ПЛК языки, не входящие в стандарт МЭК 61131-3 (например, язык С) и по каким причинам?

Ответ: не приходилось.

6. Приходилось ли вам при программировании ПЛК на языках стандарта МЭК 61131-3 использовать объектно-ориентированный подход (наследование, инкапсуляцию) и если да – то для каких задач? Что вы думаете о применении ООП в контексте программирования ПЛК?

Ответ: объектный подход и так всё время использую в функциональных блоках, перегружаю функции для того чтобы охватить все виды объектов. ООП полноценно пробовал в CODESYS V3.5 с наследованием, интерфейсами и т.д., но показалось слишком неудобным для моих не слишком сложных проектов. К тому же использую оба кодесиса (V2.3 и V3.5) в работе одинаково часто и нужен примерно одинаковый код.

7. Встречались ли вы с ситуациями, когда заказчик настаивал на использовании конкретного языка программирования? Чем были вызваны такие требования?

Ответ: иногда просят писать на CFC для того чтобы заказчик или его сотрудники могли попытаться сами разобраться и как-то изменять код.

8. Каким вы видите будущее стандарта МЭК 61131-3 и языков программирования ПЛК? Как, на ваш взгляд, будет происходить программирование ПЛК через 20 лет?

Ответ: скорее всего, слабо изменится по сравнению с другими видами программирования. Но всё идёт по пути автоматизации рутинных задач и большему применению машинного обучения. Будет здорово, если в эту сферу придут решения, когда технологу можно будет простыми средствами описать задачу и основная часть кода сгенерируется автоматически.

Не сильно изучал Automation Server от CODESYS, но, надеюсь, будут подвижки в автоматизации работы со множеством ПЛК – массовая и автоматическая загрузка обновлений кода, например. Сейчас вот обслуживаю системы (тепловые сети и фандоматы), где более 500 ПЛК с похожими и почти идентичными программами, и массовое обновление занимает несколько месяцев.

Заключение

Разумеется, строить какую-то статистику после опроса такого небольшого количества людей было бы совершенно некорректно. Тем не менее, в ответах конкретной опрошенной группы разработчиков можно увидеть определенные закономерности:

- основным языком для большинства респондентов является ST;
- большинство респондентов либо уже используют ООП в своих проектах, либо проявляют к нему интерес;
- многие респонденты сделали акцент не на особенностях МЭК-языков, а на особенностях их реализации в конкретной среде разработки; они отметили, что наличие удобной IDE является крайне важным фактором при разработке проектов;
- четверо из респондентов предполагают, что будущее программирования ПЛК связано с использованием языка Python.