ЭФФЕКТИВНОСТЬ ВСТРОЕННОГО ПРОГРАММИРОВАНИЯ ДЛЯ ПЛК

БИЛЛ ДИХНЕР (BILL DEHNER)
ПЕРЕВОД: ВЛАДИМИР РЕНТЮК

Правильный выбор соответствующего приложению контроллера и сопутствующего программного обеспечения значительно повышает эффективность процесса программирования.

Контроллер выбирают на основе той роли, которую он играет в конкретном приложении. Однако необходимо учитывать и возможности сопутствующей программной платформы для программирования контроллера, поскольку это оказывает весьма заметное влияние на временные затраты и эффективность его программирования.

Конечно, можно запустить платформу разработки, создать новый проект и написать лестничный код с нуля практически с любым пакетом программного обеспечения (ПО) контроллера. Но этот способ требует конфигурации программируемого логического контроллера (ПЛК) буквально на лету и не настолько эффективен, как другие методы. Задача упрощается, если использовать отдельные программные платформы, предназначенные для программирования контроллера и имеющие встроенную эффективность. Благодаря своим конструктивным особенностям эти платформы предоставляют разработчику оптимальный путь конфигурирования, уменьшая его усилия, необходимые для завершения программы в целом.

Для более эффективного программирования упомянутый подход содержит два метода — конфигурацию «сверху вниз» и использование концепции, ориентированной непосредственно на само физическое устройство (рис. 1).

Конфигурация «сверху вниз» обеспечивает программисту четкий понятный путь, показывая ему, что необходимо, а что нет при конфигурировании проекта ПЛК. Это выполняется на основе выбора из контекстного меню. Концепции, ориентированные на устройства, позволяют последним обрабатывать общие функции, скажем так, «за кулисами» сцены, на которой идет

РИС. 1. ▶ Некоторые ПЛК, предназначенные для более эффективного программирования, (например контроллеры семейства Do-more BRX Micro компании AutomationDirect), уже обеспечивают не только конфигурацию «сверху вниз», но и подход, ориентированный непосредственно на устройство

представление, освобождая специалиста от выполнения рутинных задач.

Примеры и объяснения конфигурации «сверху вниз» помогут увидеть повышение эффективности и понять особенности программирования, ориентированного на само физическое устройство.

КОНФИГУРАЦИЯ «СВЕРХУ ВНИЗ»

Для некоторых контроллеров конфигурация упрощается благодаря методу «сверху вниз».

Этапы конфигурации «сверху вниз»:

1. Конфигурация процессора.
2. Конфигурация портов ввода/вывода (I/O).
3. Конфигурация модуля.
4. Конфигурация устройства.
5. Отображения ввода/вывода (I/O).
6. Конфигурация памяти.

Порядок целей имеет свой приоритет, причем каждый элемент в списке зависит от элемента или элементов, расположенных над ним, т. е. сверху. Например, все, что находится ниже конфигурации центрального процессора (ЦПУ), зависит от того, как был реализован сам ЦПУ. Последовательный порт, тип порта, основные опции и параметры ввода/вывода Ethernet, а также опции и параметры сервера, в частности Modbus/TCP и EtherNet/IP[1], использующие explicit messaging (явные, подробные сообщения) на встроенном порте Ethernet, представляют собой настройки, обычно доступные во время конфигурации ЦПУ. Таким образом, добавляются необходимые параметры для определенных или всех элементов, находящихся ниже стадии конфигурации ЦПУ.

Конфигурирование контроллера в должном порядке помогает всем элементам, связанным с ним, встать на свои места, а также облегчить и автоматизировать некоторые стадии разработки всего ПО. Так, настройка ЦПУ как клиента Modbus RTU влияет, например, на элементы, расположенные ниже по рейтингу, раскрывая соответствующие опции и параметры и тем самым упрощая последующие шаги программирования.

КОНЦЕПЦИИ, ОРИЕНТИРОВАННЫЕ НА ФИЗИЧЕСКОЕ УСТРОЙСТВО

Эффективность разработки, осуществляемой методом упорядоченной конфигурации, напрямую связана с концепциями, ориентированными на физическое устройство. В рамках этих концепций лестничный код взаимодействует с устройством, находящимся как бы в середине, а не непосредственно с самим оборудованием (рис. 2). Устройство здесь аналогично драйверу принтера персонального компьютера, где драйвер (в нашем случае устройство) обрабатывает все низкоуровневые данные, поэтому программист может отправлять их на принтер, не беспокоясь о программировании самого принтера.

Программист обычно думает об устройстве как о датчике, энкодере (кодовом датчике положения рабочего органа), модуле ввода/вывода, частотно-регулируемом электроприводе (англ. Variable Frequency Drive, VFD), EtherNet/IP-модуле, удаленной стойке или подобном физическом аппаратном обеспечении. Вместо этого в контроллере, ориентированном на физическое устройство, составные части представляют собой фрагменты кода между программой и аппаратным обеспечением. С помощью рассматриваемой концепции устройство будет настроено и сможет обрабатывать подпрограммы для управления контроллером конечного оборудования — для установления протоколов связи, подтверждения связи и определения требований к памяти. Большая часть данных подпрограмм оборудования обрабатывается через конфигурацию каждого устройства, а не путем общего программирования контроллера.

ПОСЛЕ ЗАВЕРШЕНИЯ КОНФИГУРАЦИИ

Как только устройство настроено, инструкция программы общается уже с конкретным прибором, а не напрямую с оборудованием. Команда использует определенную область памяти, биты подтверждения и флаги памяти, созданные во время конфигурации. Эти инструкции также могут напрямую обращаться к памяти с помощью битов и целых чисел и извлекаться из памяти для принятия тех или иных логических решений. Например, инструкция может как некая математическая функция выполнять и возвращать результат с плавающей запятой обратно в память.

Непосредственно к памяти обращается и последовательный порт. Когда данные поступают в порт или из него, устройство обрабатывает буферизацию и флаги состояния в памяти.

В качестве подобного устройства можно рассматривать и сервер. Он функционирует в фоновом режиме, общается напрямую с аппаратным обеспечением и перемещает данные между оборудованием и памятью. Пример — Modbus TCP. В основном он работает вне программы контроллера, но к нему всегда можно получить доступ.

ПРИМЕР ПРИМЕНЕНИЯ

Независимо от сложности выбранного оборудования устройство обеспечивает понятный, единообразный интерфейс между аппаратным обеспечением и программой контроллера. Причем каждое из них настроено таким же образом, как, например, Modbus/RTU или универсальный последовательный порт, согласно инструкциям «сверху вниз», требующим лишь выбора необходимых функций и заполнения ими нескольких пробелов.

Если взять типичное приложение, в частности дивертор, предназначенный для изменения марш-

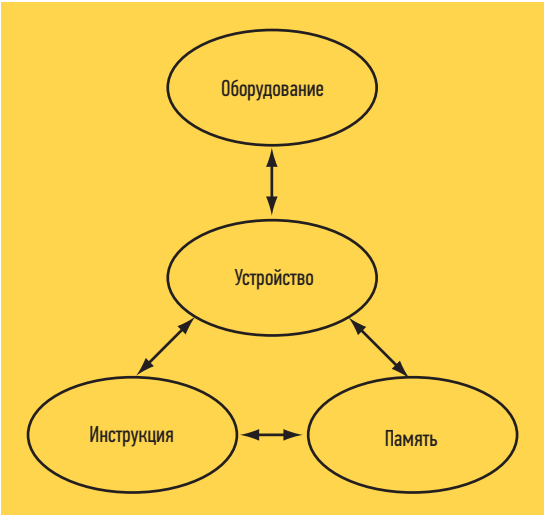

РИС. 2. ▼ В программной платформе ПО Do-more Designer, созданной компанией AutomationDirect для разработчиков программного обеспечения ПЛК средств автоматизации, применен подход, ориентированный на конечное физическое устройство

[1] EtherNet/IP — промышленный протокол Ethernet от Open DeviceNet Vendor Association (ODVA).

рута движения изделий в упаковке, он содержит целый ряд аппаратных компонентов, которыми необходимо должным образом управлять и контролировать в комплексе. Приложение может включать энкодер синхронизации затвора устройства для задания направления движения коробок с учетом различия их габаритов, электродвигатель, управляемый посредством частотного регулирования, и считыватель штрихкода для сканирования движущейся коробки для определения адреса назначения. Здесь также понадобятся входы и выходы, необходимые для мониторинга данных датчиков обнаружения коробки, а также для управления пневматическими приводами, в том числе подъемными устройствами.

Как демонстрирует это приложение на примере системы сортировки коробок, здесь нет ничего необычного в том, что к ПЛК подключено несколько различных аппаратных средств автоматизации, причем каждый аппаратный компонент и его соединения определяют такое устройство на физическом уровне. Однако в этом случае контроллер с конфигурацией «сверху вниз» и концепциями, ориентированными на устройства, настраивается быстрее, к тому же большая часть операций по настройке осуществляется автоматически.

Если контроллеру недостаточно встроенных высокоскоростных входов для подсчета квадратурных (со сдвигом на 90°) импульсов энкодера, то можно использовать высокоскоростной счетчик. Этот модуль не является частью конфигурации ЦПУ, но автоматически обнаруживается на втором этапе, при настройке ввода/вывода. Третий шаг — конфигурация самого модуля — предусматривает автоматическое изменение нужных параметров со значениями по умолчанию. На этом этапе могут быть сделаны любые коррективы в требуемой конфигурации. ПЛК автоматически обрабатывает карту ввода/вывода для добавленного модуля и создает необходимые адреса регистра, с отображением портов ввода/вывода в памяти.

Порт Ethernet контроллера связывается с частотным приводом. Начиная с вершины списка конфигурации, мастер ввода/вывода Ethernet включен как часть шага конфигурации ЦПУ. В настройках ввода/вывода создается запись, позволяя установить IP-адрес и другие параметры связи. Для конфигурации данного устройства больше ничего не требуется, поскольку остальные параметры, такие как сопоставление ввода/вывода, реализуются автоматически.

Последовательный порт контроллера используется для связи со сканером штрихкода, для чего предназначены простые текстовые строки ASCII. Порт распознается на этапе настройки ЦПУ, когда универсальный последовательный порт настроен, то включает такие параметры, как скорость передачи данных и аппаратные протоколы, например RS-232. Конфигурация портов ввода/вывода и самого модуля уже не требуется, поскольку процедура выполняется автоматически, обеспечивая предварительно сконфигурированный интерфейс с доступом к соответствующим системным ресурсам. Шаг конфигурации памяти автоматически выделяет память для этого устройства.

Многоточечные дискретные входные и выходные модули используются для контроля и управления датчиками и пневматикой. Они настроены аналогичным образом. Некоторые из них просты в настройке, другие — сложнее, но все они применяют одинаковую методологию, «сверху вниз».

ЭФФЕКТИВНОСТЬ ИНСТРУКЦИЙ

Как показано на рис. 2, контроллеры с платформой программирования «сверху вниз», ориентированные на физическое устройство, быстро конфигурируются. Данные контроллеры обычно обеспечивают и более эффективные инструкции, в том числе петли обратной связи, выполненные на основе пропорционально-интегрально-дифференциального регулирования (ПИД-регуляторы), и блоки управления движением.

Как известно, для ПИД-регуляторов существуют тысячи применений, поэтому здесь нет единого общего решения. К тому же у различных контроллеров имеются в этом плане разные возможности. И если у одних есть определенные ограничения, то у других благодаря ПИД-регулированию присутствуют независимые, модульные, взаимозаменяемые и управляемые по времени технологии, которые предназначены для решения задач конкретных приложений.

Одним из путей повышения эффективности является разбивка петли на основе ПИД-регулирования на более мелкие элементы. Скажем, вместо того чтобы объединять все параметры регулятора, такие как характеристики фильтров, масштабирование, таблицы профилей участков наклон/выдержка и обработки сигналов тревоги, в одной команде ПИД-управления, здесь для индивидуального доступа к параметрам и упрощения настройки этих алгоритмов используются отдельные команды. Эти инструкции отображаются и на дисплее, что помогает лучше понять контур управления, точнее выполнить начальные настройки и устранить возможные неполадки.

Инструкции по управлению движением могут следовать по аналогичному пути, разбивая параметры управления на разные уровни в зависимости от сложности обучения. Простые инструкции описания движения позволяют быстро применять базовые команды перемещения с минимально требуемой конфигурацией. Инструкции по перемещению на промежуточном уровне предоставляют больше пользовательских параметров. Расширенные инструкции предусматривают выбор или создание настраиваемых профилей перемещения, часто с помощью более простого процесса конфигурирования.

Новые, расширенные контроллеры упрощают программирование, чаще применяя подход «сверху вниз» и используя концепции, ориентированные на физическое устройство. При правильной конфигурации большая часть программирования интерфейса устройства оказывается между ПО контроллера и оборудованием и осуществляется автоматически и эффективно без необходимости написания кода. При этом более быстрое конфигурирование приводит к меньшим временным затратам на программирование, а кроме того, дополняется и более широким набором доступных инструкций. ●