УДК 004.942.519.876.5

ВОЗМОЖНОСТИ СИСТЕМЫ ПРОГРАММИРОВАНИЯ CoDeSys ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ПРОЕКТИРОВАНИИ МИКРОПРОЦЕССОРНЫХ СИСТЕМ УПРАВЛЕНИЯ

Инж. НОВИКОВ С. О.

Белорусский национальный технический университет

Задачи управления системами различного типа, вероятно, потерять своей актуальности не смогут ни в настоящее время, ни в будущем. Технический прогресс, совершенствование технологических баз, бурное развитие микросхемотехники требуют от систем управления сегодняшнего дня соответствующих изменений в принципах подхода к управлению и регулированию характеристик систем управления. И если на уровнях, где в решаемых задачах нет ограничений на моменты инерций и сопротивлений на валах силовых агрегатов, представлены шаговые машины с устоявшимися алгоритмами их управления, то в вопросах контроля и управления двигателями постоянного и переменного токов с переменными нагрузками на валах эти задачи становятся все более актуальными. Как очевидный факт мы принимаем системы микропроцессорного управления такими приводами, но оставляем за собой право иметь возможность изучать и исследовать поведение таких систем.

В последние годы все более широко для управления такими системами применяются комбинированные схемы программно-аппаратного управления. В их основе лежат стандартные микропроцессорные наборы, выпускаемые отечественной и зарубежной промышленностью. Возможности мик-

ропроцессорных контроллеров сегодня практически неограниченны. Ведущие западные компании разрабатывают унифицированные способы создания микроконтроллеров, возможности эмуляции вновь создаваемых систем управления в реальном масштабе времени, разработку пакетов математического сопровождения на языках низкого уровня и многие другие моменты для все более простого и менее затратного производства управляющих контроллеров.

И тем не менее не стоит забывать, что изначально микропроцессоры были созданы не столько для работы в персональных компьютерах (ПК), сколько в качестве программно-управляемых устройств для автоматизации промышленности и бытовой техники. На их основе были созданы программируемые логические контроллеры (ПЛК) – устройства, автоматизирующие работу как отдельных аппаратов, например станков с программным управлением или стиральных машин и микроволновых печей, так и огромных производственных комплексов. Таким образом, программируемые логические контроллеры – даже более распространенные устройства, чем ПК, количество которых во всем мире недавно превысило магическую цифру в 1 млрд.

В настоящее время основное внимание уделяется технологиям создания программного обеспечения для систем управления промышленной автоматикой, построенных на базе ПЛК, и практическому программированию на языках стандарта Международной электротехнической комиссии (МЭК) 61131-3.

К сожалению, период широкого распространения стандарта МЭК 61131-3 пришелся на годы перестройки. Отсутствие спроса промышленности на средства автоматизации производства привело к распаду большинства коллективов, занятых применением ПЛК, что отразилось и на уровне подготовки специалистов. В настоящее время наблюдается существенный рост потребности в современных инструментах производства и автоматики. Широкое распространение и доступность персональных компьютеров привели к появлению большого числа специалистов, профессионально владеющих компьютерными технологиями. Поэтому неудивительно, что сегодня персональные компьютеры массово применяют на всех уровнях промышленной автоматизации, включая классические контроллерные задачи. Между тем, во многих случаях применение промышленных ПК не оправдано экономически и технически сложно. Даже там, где задача на ПЛК решается «в одно действие» и на два порядка дешевле, нередко применяют дорогостоящие промышленные ПК, операционные системы реального времени и заказное программное обеспечение. Единственной причиной такого подхода является наличие подготовленных специалистов. Однобокости решений немало сопутствует и почти полное отсутствие современной литературы по применению ПЛК на русском языке. При знакомстве с доступными источниками 10-летней давности создается впечатление, что применение ПЛК связано с примитивным и неуклюжим программированием при помощи специализированных пультов, что выглядит чрезвычайно архаично.

Любая машина, способная автоматически выполнять некоторые операции, имеет в своем составе управляющий контроллер – модуль, обеспечи-

вающий логику работы устройства. Контроллер – это мозг машины. Естественно, чем сложнее логика работы машины, тем «умнее» должен быть контроллер. Технически контроллеры реализуются по-разному. Это могут быть механическое устройство, пневматический или гидравлический автомат, релейная или электронная схема или даже компьютерная программа. В случае, когда контроллер встроен в машину массового выпуска, стоимость его проектирования распределена на большое число изделий и мала в отношении к стоимости изготовления. В случае машин, изготавливаемых в единичных экземплярах, ситуация обратная. Стоимость проектирования контроллера доминирует по отношению к стоимости его физической реализации.

Физически типичный ПЛК представляет собой блок, имеющий определенный набор выходов и входов, для подключения датчиков и исполнительных механизмов (рис. 1). Логика управления описывается программно на основе микрокомпьютерного ядра. Абсолютно одинаковые ПЛК могут выполнять совершенно разные функции. Причем для изменения алгоритма работы не требуется каких-либо переделок аппаратной части. Аппаратная реализация входов и выходов ПЛК ориентирована на сопряжение с унифицированными приборами и мало подвержена изменениям.

Рис. 1. Принцип работы ПЛК

Задачей прикладного программирования ПЛК является только реализация алгоритма управления конкретной машиной. Опрос входов и выходов контроллер осуществляет автоматически, вне зависимости от способа физического соединения. Эту работу выполняет системное программное обеспечение. В идеальном случае прикладной программист совершенно не интересуется, как подсоединены и где расположены датчики и исполнительные механизмы. Кроме того, его работа не зависит от того, с каким контроллером и какой фирмы он работает. Благодаря стандартизации языков программирования прикладная программа оказывается переносимой. Это означает, что ее можно использовать в любом ПЛК, поддерживающем данный стандарт.

Изначально ПЛК предназначались для управления последовательными логическими процессами, что и обусловило слово «логический» в названии устройства. Современные ПЛК, помимо простых логических операций, способны выполнять цифровую обработку сигналов, управление приводами, регулирование, функции операторского управления и т. д.

Для ПЛК существенное значение имеет не только быстродействие самой системы, но и время проектирования, внедрения и возможной оперативной переналадки.

Абсолютное большинство ПЛК работают по методу периодического опроса входных данных (сканирования). ПЛК опрашивает входы, выпол-

няет пользовательскую программу и устанавливает необходимые значения выходов. Специфика применения ПЛК обусловливает необходимость одновременного решения нескольких задач. Прикладная программа может быть реализована в виде множества логически независимых задач, которые должны работать одновременно.

ПЛК обычно имеет один процессор и выполняет несколько задач псевдопараллельно, последовательными порциями. Время реакции на событие оказывается зависящим от числа одновременно обрабатываемых событий. Рассчитать минимальное и максимальное значения времени реакции, конечно, можно, но добавление новых задач или увеличение объема программы приведет к увеличению времени реакции. Такая модель предпочтительна для систем мягкого реального времени. Современные ПЛК имеют типовое значение времени рабочего цикла, измеряемое единицами миллисекунд и менее. Поскольку время реакции большинства исполнительных устройств значительно выше, с реальными ограничениями возможности использования ПЛК по времени приходится сталкиваться редко.

В некоторых случаях ограничением служит не время реакции на событие, а обязательность его фиксации, например работа с датчиками, формирующими импульсы малой длительности. Это ограничение преодолевается специальной конструкцией входов. Так, счетный вход позволяет фиксировать и подсчитывать импульсы с периодом во много раз меньшим времени рабочего цикла ПЛК. Специализированные интеллектуальные модули в составе ПЛК позволяют автономно отрабатывать заданные функции, например модули управления сервоприводом.

Контроллеры традиционно работают в нижнем звене автоматизированных систем управления предприятием (АСУ) – систем, непосредственно связанных с технологией производства (ТП). ПЛК обычно являются первым шагом при построении систем АСУ. Это объясняется тем, что необходимость автоматизации отдельного механизма или установки всегда очевидна. Она дает быстрый экономический эффект, улучшает качество производства, позволяет избежать физически тяжелой и рутинной работы. Контроллеры по определению созданы именно для такой работы.

Далеко не всегда удается создать полностью автоматическую систему. Часто «общее руководство» со стороны квалифицированного человека – диспетчера – необходимо. В отличие от автоматических систем управления такие системы называются автоматизированными. Еще 10–15 лет назад диспетчерский пульт управления представлял собой табло с множеством кнопок и световых индикаторов. В настоящее время подобные пульты применяются только в очень простых случаях, когда можно обойтись несколькими кнопками и индикаторами. В более «серьезных» системах применяются ПК.

Появился целый класс программного обеспечения реализующего интерфейс «человек – машина» (MMI). Это так называемые системы сбора данных и оперативного диспетчерского управления (Supervisory Control And Data Acquision System – SCADA). Современные SCADA-системы выполняются с обязательным применением средств мультимедиа. Помимо живого отображения процесса производства, хорошие диспетчерские системы позволяют накапливать полученные данные, проводят их хранение и ана-

лиз, определяют критические ситуации и производят оповещение персонала по каналам телефонной и радиосети, позволяют создавать сценарии управления (как правило, Visual Basic), формируют данные для анализа экономических характеристик производства.

Создание систем диспетчерского управления является отдельным видом бизнеса. Разделение производства ПЛК, средств программирования и диспетчерских систем привело к появлению стандартных протоколов обмена данными. Наибольшую известность получила технология OPC (OLE for Process Control), базирующаяся на механизме DCOM Microsoft Windows. Механизм динамического обмена данными (DDE) применяется пока еще достаточно широко, несмотря на то что требованиям систем реального времени не удовлетворяет.

Одной из систем для разработки программного обеспечения для ПЛК является программный комплекс CoDeSys. Он разработан компанией 3S-Smart Software Solutions GmbH (3S). Его основное назначение – программирование ПЛК и промышленных компьютеров в стандарте МЭК 61131-3. Ряд неординарных решений 3S привел к тому, что CoDeSys стал штатным инструментом программирования ПЛК ведущих европейских изготовителей: ABB, Beckhoff, Beck IPC, Berger Lahr, Bosch Rexroth, ifm, Keb, Kontron, Lenze, Moeller, WAGO, Fastwel и др. Некоторые из них используют CoDeSys как базовое ядро собственных систем программирования, известных под собственными торговыми марками.

Как средство программирования ПЛК CoDeSys можно разделить на две части: среду программирования и систему программирования. Среда программирования функционирует на персональном компьютере в среде Windows.

Поскольку CoDeSys дает машинный код, поддержка его программирования достаточно проста и по минимуму сводится к набору функций поддержки ввода-вывода и отладки. Система программирования функционирует в ПЛК и обеспечивает загрузку кода прикладной программы, «горячее» обновление кода, отладку, управление задачами и некоторые сервисные функции. Система программирования поставляется 3S изготовителям ПЛК (OEM) в виде исходных текстов. Это позволяет максимально эффективно реализовать поддержку аппаратных средств, без каких либо промежуточных механизмов. Изготовителю оборудования требуется дописать аппаратно-зависимые функции ввода-вывода, возможно, отредактировать функции поддержки канала связи на физическом уровне (через API при наличии ОС) и функции записи кода прикладной программы в ППЗУ (Flash, диск и др.). Далее он компилирует готовую систему программирования и помещает ее код в ПЗУ (или на загрузочный диск) своего ПЛК. Теперь ПЛК готов для поставки заказчикам.

Из отладочных функций CoDeSys интересен инструмент графической трассировки значений переменных. С его помощью можно проводить отладку не только ПО, но и оборудования, причем без написания программы. Весьма удобно и наличие встроенной системы визуализации, функционирующей как в инструментальной среде, так и в ПЛК (имеющем дисплей) и Web. На практике пользователи выполняют достаточно сложные проекты автоматизации в CoDeSys без необходимости приобретения SCADA.

Система программирования CoDeSys может работать как поверх операционной системы, так и на «голом железе». Естественно во втором случае адаптация несколько сложнее, поскольку приходится писать собственный начальный загрузчик, системные тесты, обработчик таймера, т. е. типовую минимальную поддержку аппаратуры. Но на 8- и 16-разрядных платформах это дает существенный выигрыш по быстродействию и необходимым ресурсам. Существуют четыре разновидности систем программирования CoDeSys: CSP8, CSP16, CSP32E и CSP32F. Они предназначены для 8-, 16-, 32-разрядных процессоров без ОС соответственно. Система CSP32F ориентирована на 32-разрядные платформы с ОС РВ. Она опирается на механизмы вытесняющей многозадачности ОС, первые 3 включают собственный монитор многозадачности (без вытеснения). Пользователь может создавать многозадачные проекты даже в ПЛК, построенном на 8051.

В настоящее время CoDeSys поддерживает семейства: Intel 8051, Intel 80x86 / 80186 / Pentium, семейство ARM, MIPS, Motorola MC68000 / MC68332 / ColdFire, PowerPC, Hitachi SH 2/3/4, H8, Infineon C16x, Infineon TriCore и Texas Instruments TMS32028x.

Особняком стоит система программирования CoDeSys SP RTE для Windows XP/NT/2000. Она включает в себя собственное ядро жесткого реального времени, функционирующее под Windows. Эта система не требует адаптации. Связь с оборудованием происходит через драйверы. SDK включен в комплект поставки.

Технические преимущества CoDeSys с аналогичными системами разработки и моделирования:

- рекордное быстродействие прикладных МЭК программам. Встроенный компилятор непосредственно формирует машинный код для целевого микропроцессора. Пользователь может писать даже обработчики аппаратных прерываний на МЭК языках (если это разрешено изготовителем ПЛК) и синхронизировать параллельные процессы с микросекундной точностью;
- для создания библиотек быстрых функций не обязательно использовать внешние средства (С-компилятор). Как результат – переносимость и нормальная работа в режиме эмуляции;
- возможность включения собственных программных расширений наравне со стандартными, включая встроенную систему подсказки;
- доступны исходные тексты системы программирования;
- полная и даже расширенная реализация стандарта МЭК 61131-3;
- упрощенный SFC, CFC;
- интегрированные средства эмуляции и визуального моделирования;
- встроенные конфигураторы модульных систем и распределенных fieldbus систем (CANopen, DeviceNet, Profibus и др.);
- три механизма создания распределенных приложений: высокоуровневое связывание переменных (network variables), объектная master-slave модель (аналогично PDO в CANopen), библиотеки нестандартных сетей или распределенный ввод/вывод (реализуется изготовителем ПЛК);
- встроенная, целевая и web визуализация;
- ряд дополнительных возможностей (библиотеки CANopen, Modbus, система управления версиями ENI. SoftMotion).

Допустим, вы написали и отладили автономный проект на контроллере при помощи системы подготовки программ CoDeSys. Как теперь нужно доработать программу, чтобы связать ПЛК с системой диспетчерского управления, базой данных или Интернет-сервером? Ответ: никак. Никакого программирования далее вообще не потребуется. В комплекс программирования ПЛК входит OPC-сервер.

Он умеет получать доступ к данным ПЛК также прозрачно, как и отладчик. Достаточно обеспечить канал передачи данных ПЛК–OPC-сервер. Обычно такой канал уже существует, вы использовали его при отладке. Вся дальнейшая работа сводится к определению списка доступных переменных, правильной настройке сети, конфигурированию OPC-сервера и SCADA-системы. В целом операция очень напоминает настройку общедоступных устройств локальной сети ПК.

Вторая задача – интеграция нескольких ПЛК с целью синхронизации их работы. Здесь появляются сети, обладающие рядом специфических требований. В целом эти требования аналогичны требованиям к ПЛК: режим реального времени, надежность в условиях промышленной среды, ремонтопригодность, простота программирования. Такой класс сетей получил название промышленных сетей (fieldbus). Существует масса фирменных реализаций и достаточно много стандартов таких сетей (Bitbus, Modbus, Profibus, CANopen, DeviceNen), позволяющих интегрировать аппаратуру различных фирм, но ни один из них нельзя признать доминирующим.

Благодаря продуктивному развитию средств сетевой интеграции появилась возможность создания распределенных систем управления. В 80-е гг. XX в. доминировали ПЛК с числом в несколько сотен входов-выходов. В настоящее время большим спросом пользуются микроПЛК с количеством входов-выходов до 64. В распределенных системах каждый ПЛК решает локальную задачу. Задача синхронизации управления выполняется компьютерами среднего звена АСУ. Распределенные системы выигрывают по надежности, гибкости монтажа и простоте обслуживания.

Задачи управления требуют непрерывного циклического контроля. В любых цифровых устройствах непрерывность достигается за счет применения дискретных алгоритмов, повторяющихся через достаточно малые промежутки времени. Таким образом, вычисления в ПЛК всегда повторяются циклически. Одна итерация, включающая замер, обсчет и выработку воздействия, называется рабочим циклом ПЛК. Выполняемые действия зависят от значения входов контроллера, предыдущего состояния и определяются пользовательской программой.

По включению питания ПЛК выполняет самотестирование и настройку аппаратных ресурсов, очистку оперативной памяти данных (ОЗУ), контроль целостности прикладной программы пользователя. Если прикладная программа сохранена в памяти, ПЛК переходит к основной работе, которая состоит из постоянного повторения последовательности действий, входящих в рабочий цикл.

Рабочий цикл ПЛК состоит из нескольких фаз:

1. Начало цикла.
2. Чтение состояния входов.
3. Выполнение кода программы пользователя.
4. Запись состояния выходов.

5. Обслуживание аппаратных ресурсов ПЛК.
6. Монитор системы программирования.
7. Контроль времени цикла.
8. Переход на начало цикла.

В самом начале цикла ПЛК производит физическое чтение входов. Считанные значения размещаются в области памяти входов. Таким образом, создается полная одномоментная зеркальная копия значений входов.

Далее выполняется код пользовательской программы. Пользовательская программа работает с копией значений входов и выходов, размещенной в оперативной памяти. Если прикладная программа не загружена или остановлена, то данная фаза рабочего цикла, естественно, не выполняется. Отладчик системы программирования имеет доступ к образу входов-выходов, что позволяет управлять выходами вручную и проводить исследования работы датчиков.

После выполнения пользовательского кода физические выходы ПЛК приводятся в соответствие с расчетными значениями (фаза 4).

Обслуживание аппаратных ресурсов подразумевает обеспечение работы системных таймеров, часов реального времени, оперативное самотестирование, индикацию состояния и другие аппаратно-зависимые задачи.

Монитор системы программирования включает в себя большое число функций, необходимых при отладке программы и обеспечении взаимодействия с системой программирования, сервером данных и сетью. В функции системы программирования обычно включается: загрузка кода программы в оперативную и электрически перепрограммируемую память, управление последовательностью выполнения задач, отображение процесса выполнения программ, пошаговое выполнение, обеспечение просмотра и редактирования значений переменных, фиксация и трассировка значений переменных, контроль времени цикла и т. д.

Пользовательская программа работает только с мгновенной копией входов. Таким образом, значения входов в процессе выполнения пользовательской программы не изменяются в пределах одного рабочего цикла. Это фундаментальный принцип построения ПЛК сканирующего типа. Такой подход исключает неоднозначность алгоритма обработки данных в различных его ветвях. Кроме того, чтение копии значения входа из ОЗУ выполняется значительно быстрее, чем прямое чтение входа. Аппаратно чтение входа может быть связано с формированием определенных временных интервалов, передачей последовательности команд для конкретной микросхемы или даже запросом по сети.

Если заглянуть глубже, то нужно отметить, что не всегда работа по чтению входов полностью локализована в фазе чтения входов. Например, АЦП обычно требуют определенного времени с момента запуска до считывания измеренного значения. Часть работы системное программное обеспечение контроллера выполняет по прерываниям. Грамотно реализованная система программирования нигде и никогда не использует пустые циклы ожидания готовности аппаратуры. Для прикладного программиста все эти детали не важны. Существенно только то, что значения входов обновляются автоматически исключительно в начале каждого рабочего цикла.

Общая продолжительность рабочего цикла ПЛК называется временем сканирования. Время сканирования в значительной степени определяется

длительностью фазы кода пользовательской программы. Время, занимаемое прочими фазами рабочего цикла, практически является величиной постоянной. Для задачи среднего объема в ПЛК с системой программирования CoDeSys время распределится примерно так: 98 % – пользовательская программа, 2 % – все остальное.

Существуют задачи, в которых плавающее время цикла существенно влияет на результат, например это автоматическое регулирование. Для устранения этой проблемы в развитых ПЛК предусмотрен контроль времени цикла. Если отдельные ветви кода управляющей программы выполняются слишком быстро, в рабочий цикл добавляется искусственная задержка. Если контроль времени цикла не предусмотрен, подобные задачи приходится решать исключительно по таймерам.

Инженер, спроектировавший машину, должен иметь возможность самостоятельно написать программу управления. Никто лучше его не знает, как должна работать данная машина. Инженер, привыкший работать с электронными схемами, гораздо легче сможет выражать свои мысли в LD или FBD. Если он знаком с языками PASCAL или C, то использование языка ST не составит для него сложности.

За время развития ПЛК размер средней программы возрос более чем в 100 раз. Многие решения, требовавшие раньше аппаратной поддержки, реализуются сегодня программно. Соответственно требования к качеству программного обеспечения очень высоки. Поэтому сложную программу должны писать специалисты. Но для ответственных проектов очень важно, чтобы программа алгоритма была понятна техническому персоналу, осуществляющему настройку, сопровождение и ремонт оборудования. Они не обязаны изучать программу досконально, но понимать, что происходит, безусловно, должны.

ВЫВОД

Внедрение стандарта послужит фундаментом для создания единой школы подготовки специалистов. Человек, прошедший обучение по программе, включающей стандарт МЭК 61131, сможет работать с ПЛК любой фирмы. В то же время, если он имел ранее опыт работы с любыми ПЛК, его навыки окажутся полезными и существенно упростят изучение новых возможностей.

ЛИТЕРАТУРА

1. http://www.prolog-PLC.ru
2. http://www.3s-software.com

Представлена кафедрой ПОВТ и АС Поступила 12.12.2008