Г. К. Смолин, А. Р. Ханбеков

УПРАВЛЯЕМОЕ МГД-УСТРОЙСТВО

В современном обществе становится все более актуальным применение высоких технологий на производстве. Сегодня важное значение имеют высокая скорость получения и обработки информации в реальном времени, изменение режимов работы технологических процессов в зависимости от принятых данных, мгновенный обзор всей требуемой информации и т. д. Например, для повышения качества металла и увеличения темпов его производства необходимы внедрение и разработка автоматизированных систем управления (АСУ). Так, например, для перекачивания жидкого металла используются МГД-насосы, а управление ими часто осуществляют операторы, визуально наблюдающие процесс разлива. Нет ни сбора, ни обработки информации, нет и мониторинга процессов, протекающих внутри МГД-насосов. В современных технологиях все эти процессы должны обрабатываться в реальном времени операционной системой, а контроль над поступающими данными должна брать на себя программная часть АСУ. Сегодня в мире тысячи приложений в области промышленной автоматизации работают под управлением операционной системы *Windows*. Системы на основе персонального компьютера успешно собирают данные, управляют технологическими процессами, обеспечивают интерфейс оператора и передачу информации по сети в реальном времени, т. е. без отставания от течения управляемых процессов. Однако существует класс задач, где детище компании *Microsoft* до сих пор чувствует себя, прямо скажем, неуверенно. Это системы так называемого жесткого реального времени (*hard real-time*). Основное отличие жесткого реального времени (РВ) от мягкого (*soft real-time*) заключается в гарантированной скорости реакции системы управления на изменение внешних условий. В случае мягкого реального времени под управлением *Windows* время выполнения цикла может отличаться от итерации к итерации на сотни миллисекунд. Это не удивительно, поскольку программа делит время центрального процессора с другими и *Windows* не может гарантировать стабильную скорость отклика на внешнее воздействие (например, изменение входного сигнала). Традиционные решения в этой области лежат в плоскости специализированных систем жесткого РВ, использующих различные типы процессоров

и операционные системы совместно с соответствующими средствами разработки. Программное обеспечение *LabVIEW RT* (*Real-Time*) и платы серии *RT DAQ* (*Real-Time Data Aqusition*) позволяют выполнять требования жесткого РВ в рамках обычного *Windows*. *LabVIEW RT* расширило сферу применения популярного пакета. *LabVIEW RT* работает в обычном *Windows* и внешне практически ничем не отличается от пакета графического программирования *LabVIEW*. Пакет обладает такими преимуществами графического программирования, как скорость, простота разработки программ и тесное взаимодействие с широким набором устройств ввода/вывода сигналов. Единственное отличие заключается в возможности загружать код программы для выполнения на отдельном процессоре (ядро РВ), расположенном на плате ввода/вывода сигналов *RT DAQ*. Загрузка может осуществляться автоматически при запуске прикладной программы или вручную из меню в среде разработки *LabVIEW RT*. Однако разработка приложений в данном пакете осуществляется не совсем обычным способом.

Традиционное программирование включает в себя составление списка инструкций, выполняемых компьютером в установленной последовательности (в порядке появления в списке). Часто готовность данных определяет установленный командам порядок выполнения. Например, для выполнения команды 3 требуются данные, вычисленные командой 2 (таблица). Поэтому команда 2 должна быть осуществлена раньше команды 3. Команда 2, в свою очередь, зависит от данных команды 1. Поскольку команда 4 не зависит от результатов выполнения команд 1, 2 или 3, то время ее реализации может быть любым.

Последовательность команды

№ команды	Команда
1-я	Сложить числа А и Б
2-я	Прибавить В к результатам сложения А и Б
3-я	Разделить результат второго действия на число 3
4-я	Вычесть А из В

Этот анализ взаимозависимости данных приводит к новой идее программирования. Если определить операции и зависимости данных, компьютер может выполнять команды в любом порядке, даже параллельно, при этом сохраняя лишь зависимости по данным. Теперь всего лишь необхо-

дим способ определения зависимостей по данным. Если представить необходимые операции в виде модулей (блоков) и соединить эти блоки, чтобы показать их взаимные зависимости, то тем самым программируется выполнение операций. Схемы более наглядны и понятны, чем списки команд.

Программирование в среде *LabVIEW* и заключается в составлении именно таких схем, которые определяют зависимости данных. Среда программирования *LabVIEW* включает в себя большой набор модулей, которые позволяют производить ту или иную операцию, и инструмент соединения, который связывает эти блоки друг с другом. В качестве примера рассмотрим операцию умножения двух чисел и отображение результата (рис. 1).

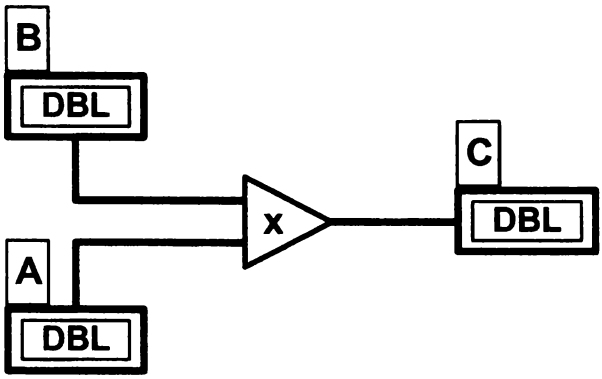

Рис. 1. Умножение двух чисел и отображение результата

Программа на языке *LabVIEW* называется виртуальным прибором (*Virtual Instrument–VI*) и является системой с двумя рабочими пространствами (окнами). Непосредственно команды находятся в одном окне, а пользовательский интерфейс (элементы управления и индикации) – в другом окне. Первое окно – программное. Это рабочее пространство блок-схемы. Второе – интерфейсное. Это окно лицевой панели, в которое пользователи вводят данные и получают выходные значения. На рис. 1 показана блок-схема обычной программы.

Язык «*G*» является «родным» языком программирования в *LabVIEW*. Некоторые аргументы могут быть приведены в пользу того, что исходный *LabVIEW*-код должен называться исходным G-кодом. Поэтому следует

сказать, что *LabVIEW* – это интегрированная среда разработки, в то время как *G* является кодом, созданным в среде разработки *LabVIEW*. Можно добавить, что графический язык «*G*» не может быть реинтерпретирован в код тексто-ориентированного языка. Вы не можете накрыть колпаком код, сгенерированный *LabVIEW*, и посмотреть спрятанный в нем текстовый код, потому что такового просто нет! Тем не менее *G* остается исходным кодом, с которым и работают программисты на *LabVIEW*.

Однако применение программного обепечения *LabVIEW* в среде *Windows* обусловлено ее большой распространенностью и доступностью для понимания. Исходные коды подходят для работы в *Windows* 9*x*/2000/*NT*, *Macintosh*, *PowerMax OS*, *Solaris*, *HP Unix*, *Sun*, *Linux*, *Pharlap RTOS* (операционная система РВ). Исходный код, разработанный для данных платформ, может быть перенесен на любую другую платформу, откомпилирован и запущен.

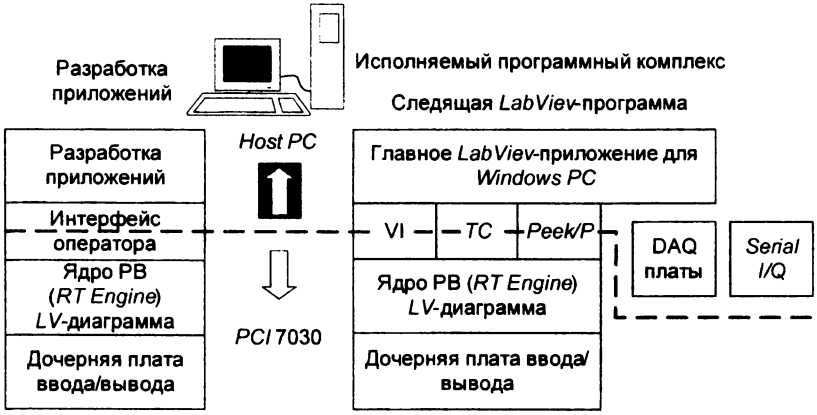

Рис. 2. Ведущее «встроенное приложение»

Такой подход позволяет гибко создавать комплексные системы измерений и автоматики МГД-устройств, где основной *Windows*-компьютер (*host PC*) выполняет основную часть задач, а встроенная плата *RT DAQ* реализует критичные ко времени приложения. Например, в среде *Windows* *LabVIEW*-программа обеспечивает интерфейс оператора, взаимодействует с аппаратурой, компьютерной сетью и сохраняет данные на диске (рис. 2).

Одновременно встроенный *LabVIEW RT*, запущенный на плате серии *RT DAQ*, осуществляет *PID*-управление в жестком РВ и обменивается данными с *Windows*-приложением. Даже в случае перезагрузки *Windows*-программы процесс *PID* регулирования не прервется ни на мгновение. После перезагрузки *Windows*-приложения соединение с работающей программой управления будет восстановлено автоматически.

Е. Д. Тельманова

ИССЛЕДОВАНИЕ ПЕРЕХОДНЫХ ПРОЦЕССОВ В СИСТЕМАХ ЭЛЕКТРОСНАБЖЕНИЯ С ПОМОЩЬЮ ДИНАМИЧЕСКИХ МОДЕЛЕЙ

Электроснабжение предприятий большой мощности имеет сложную структуру. Наряду с питанием от энергосистемы через понижающие трансформаторы, сети высшего напряжения снабжаются энергией от генераторов собственной электростанции. Комплексный характер такой системы электроснабжения усложняет расчет токов короткого замыкания, а значит, выбор и проверку электрооборудования по условию электродинамической стойкости. Итогом такого расчета является ударный ток, представляющий собой наибольшее возможное, мгновенное значение тока короткого замыкания. Согласно формуле $i_{уд} = 1{,}4k_{уд}I_{по}$, величина ударного тока прямо пропорциональна периодической составляющей в начальный момент, если она неизменна в течение всего переходного режима короткого замыкания. При этом индуктивное сопротивление цепи короткого замыкания, отнесенное к номинальному сопротивлению генератора, $X_*=X/X_{ном} \geqslant 3$.

Если это условие не выполняется, то периодическая составляющая тока короткого замыкания будет изменяться по сложному закону от начального значения I'' до установившегося значения I_∞. В этом случае при расчете токов короткого замыкания необходимо учесть переходные процессы, происходящие в генераторе.