КОМПЬЮТЕРНЫЕ НАУКИ

УДК 681.5; 535.32

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ УНИВЕРСАЛЬНОЙ СИСТЕМЫ АВТОМАТИЗАЦИИ ЭКПЕРИМЕНТА

ВЛАСЕНКО А.С., ГАЛУЗА А.А., ЗАХАРЧЕНКО А.Я.

Рассматривается проблема разработки ПО для автоматизации лабораторных установок при использовании операционных систем семейства Windows NT. Разрабатываемое ПО ориентировано на работу в составе универсальной системы автоматизации эксперимента, аппаратная часть которой рассматривалась ранее. Описываются назначение, состав, структура и особенности реализации программной части системы автоматизации.

1. Введение

В последнее время отечественные научные лаборатории все чаще сталкиваются с проблемой морального устаревания экспериментальной базы, что часто связано, прежде всего, с отсутствием автоматизации различных процессов. Покупка нового оборудования, в свою очередь, является сложной задачей по причине недостатка финансирования. При этом метрологические параметры современных приборов не намного превосходят старые аналоги. В связи с этим для поддержания экспериментальной базы на должном уровне экспериментаторы вынуждены проводить автоматизацию оборудования своими силами и средствами.

Настоящая работа является продолжением [1], где описывается аппаратная часть системы автоматизации. Основными достоинствами аппаратного решения являются простота, дешевизна и универсальность. Следует подчеркнуть, что описываемая система предназначена не для применения в серийных промышленных установках, для этого ее аппаратная часть слишком «неинтеллектуальна».

Основная цель работы – дать возможность ученому-экспериментатору в кратчайшие сроки и с минимальными финансовыми затратами автоматизировать конкретный эксперимент.

2. Постановка задачи

Для достижения поставленной цели необходимо было решить следующие задачи:

1. Создать аппаратную платформу для управления исполнительными механизмами и устройствами замера величин (решена в работе [1]).

2. Разработать и реализовать промежуточный слой программного обеспечения, который бы позволил унифицировать работу с аппаратной частью, скрыть от конечного пользователя все низкоуровневые операции и предоставить стандартный интерфейс управления системой автоматизации независимо от ее конфигурации.

3. Создать клиентское приложение для обработки полученных данных и высокоуровневой настройки конкретной реализации системы автоматизации.

Несмотря на уникальность каждой экспериментальной задачи, многие из них заключаются в измерении некоторого набора выходных величин как функции набора входных (рис. 1). Выходной величиной часто является напряжение, ток или она может быть к ним легко преобразована с помощью соответствующих датчиков (например, фотодетекторы или датчики Холла). Входные параметры задаются в виде положения некоторых регуляторов, управляющих воздействием на исследуемый или управляемый объект (барабан длин волн в оптических приборах, положение клапанов).

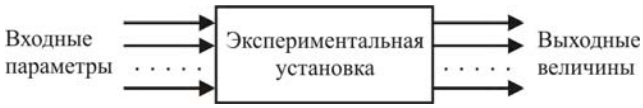

Рис. 1. Общая схема экспериментальных задач

Таким образом, для экспериментальных установок, соответствующих описанной схеме, система автоматизации должна обеспечивать автоматическое выполнение следующих шагов:

1. Замер необходимых величин.
2. Передача значений измеряемых величин в ЭВМ.
3. Обработка полученных данных.
4. Выдача команды на исполнительные механизмы (например, шаговые двигатели), изменяющие положения соответствующих регуляторов.

Ранее была разработана аппаратная часть универсальной системы автоматизации (УСА) [1], которая может быть без принципиальных изменений использована для автоматизации широкого спектра экспериментальных установок. Опишем ее вкратце. Принципиальная схема УСА приведена на рис. 2.

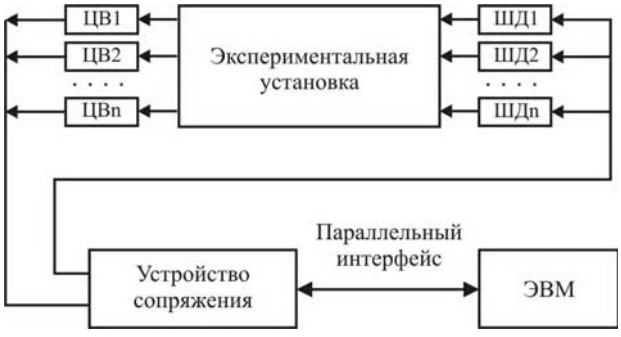

Рис. 2. Принципиальная схема универсальной системы автоматизации эксперимента: ЦВ – цифровой вольтметр; ШД – шаговый двигатель

УСА работает под управлением стандартного персонального компьютера и связана с ЭВМ при помощи параллельного интерфейса. Это вызвано простотой, синхронностью параллельного интерфейса, а также тем, что он хоть и является устаревающим, однако имеется на всех современных ЭВМ. Синхронность позволяет сделать устройство сопряжения пассивным, а следовательно, принципиально упростить его.

В качестве исполнительных механизмов УСА используются шаговые двигатели (ШД), как наиболее точные и удобные в управлении. Измерительным устройством может служить практически любой цифровой вольтметр (ЦВ), у которого имеется цифровой вывод информации. Устройство сопряжения - единственный нестандартный элемент системы – состоит из двух независимых блоков: блока коммутации с вольтметром и блока управления двигателями. При использовании параллельного интерфейса возникает проблема его ограниченной разрядности. Решением является использование параллельно-последовательного преобразователя, который и служит в качестве блока коммутации компьютера с вольтметром. Его назначение – преобразование параллельного кода результатов измерений в последовательный для ограничения используемых входных регистров порта. Проблема ограниченной разрядности возникает также при управлении шаговыми двигателями, поскольку суммарное количество обмоток всех двигателей может быть достаточно велико. Блок коммутации с двигателями позволяет ограничить число используемых выходных регистров параллельного порта, независимо от количества двигателей в установке. Вторым его назначением является усиление маломощных сигналов, поступающих из ЭВМ, перед их непосредственным попаданием на обмотки двигателей [1].

Такая организация позволяет очень просто адаптировать устройство сопряжения к различному количеству измерительных каналов и каналов управления. Делается это путем простого добавления однотипных плат с соответствующей разводкой разъемов. При этом в системе могут использоваться одновременно различные типы вольтметров и двигателей.

Данная схема уже была успешно реализована для автоматизации процессов измерений различных экспериментальных установок, таких как: спектральный криогенный эллипсометр [2] (3 двигателя и 2 вольтметра), спектрограф ИСП-52 (1 двигатель и 1 вольтметр), спектрометр-рефлектометр (1 двигатель и 2 вольтметра), инфракрасный спектрометр ИКС-31 (2 двигателя и 1 вольтметр) и др.

Итак, различные экспериментальные установки могут иметь разное количество составляющих элементов – вольтметров и шаговых двигателей (рис. 2). Более того, каждый из составляющих элементов может иметь свои параметры. Во-первых, это функциональные характеристики устройств (количество обмоток двигателей, время, необходимое для проведения измерений вольтметром и многие другие). Во-вторых – параметры взаимодействия устройств с ЭВМ – соответствие регистров параллельного порта конкретным выполняемым функциям (например, подача синхроимпульсов к двигателям). Данные параметры могут отличаться в каждой конкретной экспериментальной задаче. Таким образом, программное обеспечение должно быть максимально гибким и универсальным для обеспечения работы с экспериментальными установками любой конфигурации.

Как было сказано ранее, УСА связана с ЭВМ через параллельный интерфейс. Модель порта параллельного интерфейса приведена на рис. 3.

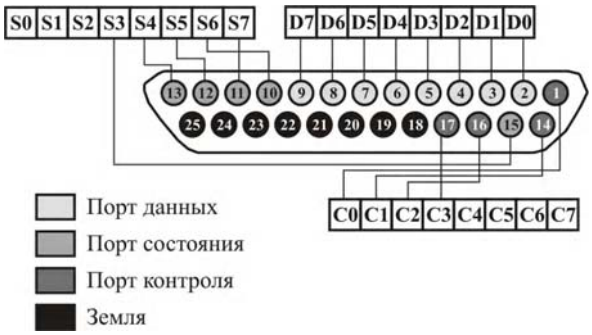

Рис. 3. Программная модель порта параллельного интерфейса

Из рис.3 видно, что программно LPT-порт разделен на три части: порт данных, порт состояния, порт управления. При этом порт данных позволяет производить подачу сигналов в устройство со всех восьми разрядов при однонаправленном режиме, а также производить запись и чтение в остальных случаях. У порта состояния доступно пять разрядов для чтения; порт контроля, в свою очередь, позволяет использовать для записи только четыре разряда [6]. Каждый из разрядов порта состояния и контроля имеет свое назначение при использовании параллельного порта для работы принтера. Однако в общем случае возможно произвольное использование данных регистров по своему назначению.

Таким образом, следующим этапом в разработке УСА является создание универсального программного обеспечения (ПО).

3. Требования к программному обеспечению УСА

Так как система связана с ЭВМ при помощи параллельного интерфейса, в первую очередь возникает необходимость создания драйвера режима ядра (рис. 4), позволяющего осуществлять управление LPT-портом компьютера. Данная необходимость вызвана тем, что операционные системы семейства Windows NT не допускают использования привилегированных инструкций ввода-вывода, необходимых для низкоуровневого общения с портом, в коде приложений пользовательского режима [3]. При этом стандартные драйверы LPT-порта допускают обращение к порту лишь с общими запросами по чтению-

записи данных. Возникает необходимость проникновения кода программы в режим ядра операционной системы, что и должен обеспечить драйвер. Этот драйвер должен быть универсальным, т.е. управлять параллельным портом без привязки к какому-либо конкретному варианту реализации системы автоматизации.

Каждая экспериментальная задача уникальна и требует от пользовательского приложения специфических действий. С другой стороны, логика работы с универсальной системой автоматизации не зависит от конкретной задачи. Поэтому для упрощения разработки пользовательского приложения необходимо создать промежуточный слой программного обеспечения, который представляет собой библиотеку классов, предоставляющую стандартный интерфейс для работы с универсальной системой автоматизации. Использование промежуточного слоя является стандартным решением в современных программных технологиях (например, DirectX, OpenGL [7]). Этот слой играет роль своеобразного посредника между кодом драйвера и клиентской программой. Классы библиотеки должны описывать объекты реального мира (например, цифровой вольтметр или экспериментальную установку в целом) и своими свойствами и методами предоставлять интерфейс для работы с устройством клиентскому приложению. Библиотека должна содержать в себе все настройки, которые понадобятся для управления конкретными реализациями устройства, т.е. настройки работы драйвера. Таким образом, данную библиотеку можно также считать драйвером пользовательского режима.

Третья составляющая программного обеспечения – клиентская программа (см.рис. 4), которая и будет конечным пунктом в движении результатов экспериментов от устройства к пользователю. Клиентское приложение не должно управлять напрямую аппаратной установкой или системным драйвером, а взаимодействовать только с объектами классов, описанных в упомянутой выше библиотеке. Так как аппаратная часть системы автоматизации является пассивным устройством и неспособна сообщать ЭВМ о своих характеристиках (как это обычно происходит во время установки различных устройств ЭВМ), в клиентском приложении необходимо предусмотреть удобную схему выбора конфигурации устройства, а значит определения настроек работы драйверов обоих уровней. Таким образом, скрытием всей низкоуровневой логики работы с устройством, а также системой гибкой настройки достигается универсальность клиентского приложения.

Учитывая все сказанное, ПО универсальной системы автоматизации эксперимента должно включать в себя драйвер режима ядра, библиотеку классов и клиентское приложение (см.рис. 4).

Рис. 4. Составляющие программного обеспечения системы автоматизации

4. Структура и взаимодействие ПО

Как видно из рис.4, программное обеспечение имеет многослойную структуру, что позволяет каждой составляющей части заботиться о выполнении работы только на своем уровне абстракции. При этом, естественно, слои связаны между собой. Обмен данными между клиентским приложением и библиотекой классов осуществляется путем передачи параметров при вызовах соответствующих методов объектов (экземпляров классов). Связь между библиотекой классов и системным драйвером режима ядра происходит за счет вызова системных функций для работы с драйверами. При этом в силу многослойности операционной системы Windows порождается немалое количество различных промежуточных вызовов (вызовы функций подсистемы среды, вызовы системных сервисов, работа менеджера ввода-вывода (рис. 5), что в конечном итоге позволяет передавать данные в код ядра операционной системы [4].

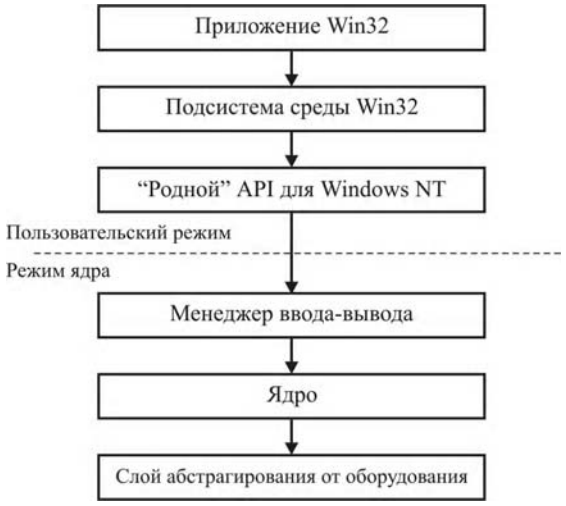

Рис. 5. Взаимодействие системных программных слоев в Windows NT

На рис. 5 показано взаимодействие различных программных слоев в операционных системах Windows NT. В нашем конкретном случае взаимодействие библиотеки классов и системного драйвера

данную схему можно расшифровать так. Приложение Win32 (библиотека классов) вызывает API Win32-функцию для взаимодействия с драйвером. Данная функция, находящаяся в клиентской библиотеке kernel32.dll подсистемы среды Win32, вызывает необходимую функцию библиотеки ntdll.dll, которая затем передает управление одному из системных сервисов (в данном случае менеджеру ввода-вывода), который и взаимодействует с драйвером [4]. Слой абстрагирования от оборудования позволяет избежать использования платформенно-зависимых инструкций в коде ядра, тем самым повышая переносимость программного кода.

И, наконец, обмен данными между системным драйвером (фактически промежуточным слоем между программной и аппаратной частью ЭВМ) и аппаратной частью системы автоматизации осуществляется при помощи параллельного интерфейса.

5. Описание ПО системы автоматизации

На данный момент нами разработаны драйверы двух уровней – режима ядра и пользовательского режима, которые уже можно использовать как для простого общения с LPT-портом компьютера, так и для выполнения всех основных операций при работе с аппаратной частью системы автоматизации. Также разработано простое клиентское приложение, позволяющее производить настройку работы с устройством и предоставляющее основные возможности по взаимодействию с экспериментальной установкой.

LPTPort.sys является типичным драйвером режима ядра.Он монолитный, так называемый «драйвер в стиле NT» [3]. Это означает, что драйвер управляет процессом общения устройства от поступления запроса пользовательского приложения (в данном случае библиотеки классов) до фактического вывода данных на регистры LPT-порта. Несмотря на то, что данная архитектура драйверов является более старой в отличие от многослойной архитектуры Windows Driver Model, она целиком подходит для этой задачи, а также более проста в реализации. Все это и определило ее выбор.

Другим важным фактором, определяющим специфику данного драйвера, является пассивность аппаратной части системы автоматизации, т.е. неспособность генерации прерываний устройством. При этом связь с аппаратной частью является однонаправленной – устройство неспособно сообщить ЭВМ о каких-либо событиях (окончание измерения вольтметра, поворот шагового двигателя и т.д.). Как следствие, для драйвера была выбрана политика выполнения задержек, необходимых для корректной работы устройства (например, время для осуществления измерений цифровым вольтметром).

Драйвер режима ядра выполняет основную логику работы с устройством, не зависящую от его конкретной реализации – запись данных в регистры порта, считывание результатов измерений в виде цепочки бит.

Драйвер пользовательского режима DirectLPT.dll является типичной динамической библиотекой [5]. Однако она содержит в своем составе не просто множество функций, а набор классов с их свойствами и методами.

Библиотека содержит основной класс LPTExternalDevice и набор структур, хранящих различные настройки текущего устройства. Объект класса LPTExternalDevice является моделью реального физического устройства – экспериментальной установки. Данный класс помимо прочих свойств содержит в своем составе структуру DEVICE_DESCRIPTION, которая хранит в себе все настройки, необходимые как для работы непосредственно библиотеки, так и для работы системного драйвера (рис. 6). Именно эта структура передается библиотеке из клиентской программы во время инициализации драйверов. Некоторые данные в DEVICE_DESCRIPTION сгруппированы в отдельные структуры. Так, можно выделить массивы структур DEVICE_ENGINE и DEVICE_VOLTMETER, которые хранят настройки, необходимые для работы шаговых двигателей и цифровых вольтметров системы автоматизации.

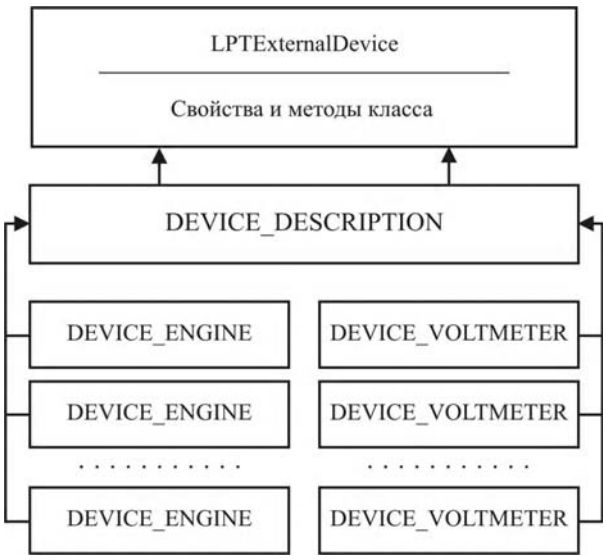

Рис. 6. Общая структура класса LPTExternalDevice

Как уже было сказано, различные реализации аппаратных частей системы автоматизации могут иметь различные параметры, начиная от простого расположения регистров порта, отвечающих за те или иные функции, заканчивая разным количеством управляемых элементов (шаговых двигателей и вольтметров). Как следствие – работа драйверов обоих уровней должна быть максимально гибкой, без привязки к каким-либо постоянным значениям. В свою очередь, клиентское приложение должно обеспечивать механизм настройки всех параметров работы с устройством. При внесении любых изменений в конфигурацию аппаратной части, а также в начале работы приложения происходит инициализация драйверов. Про-

цесс инициализации драйверов обоих уровней показан на рис. 7. Структура DEVICE_DESCRIPTION, содержащая настройки, передается библиотеке, где сохраняется в соответствующем поле класса. Часть этих параметров используется непосредственно кодом библиотеки DirectLPT.dll. Это настройки по выполнению высокоуровневых действий при работе с устройством (например, расшифровка данных измерений вольтметра). Другая же часть передается далее в код драйвера, где сохраняется в так называемой структуре расширения объекта устройства (определяемая пользователем структура, которая создается перед началом работы драйвера; именно в этой структуре принято хранить основные данные взамен использования глобальных переменных) [3]. Эти настройки используются для работы драйвера: значения задержек, карты регистров порта.

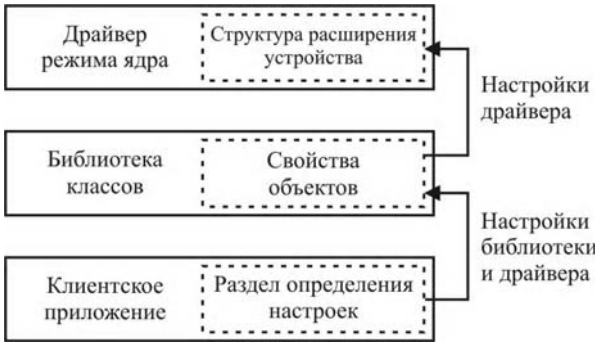

Рис. 7. Схема инициализации драйверов

6. Заключение

Научная новизна работы состоит в формулировке единой системы требований к функциональности аппаратного и программного обеспечения, которым должна удовлетворять система автоматизации для возможности ее использования в широком спектре лабораторных экспериментов.

Практическое значение разработанной и реализованной программно-аппаратной системы заключается в возможности автоматизации широкого спектра экспериментальных установок за короткое время. При этом затраты средств и усилий на установку, настройку и управление как аппаратной, так и программной частью системы минимальны.

На данный момент клиентское приложение имеет достаточно простую конфигурацию. Оно выполняет лишь основные действия по взаимодействию с экспериментальной установкой (управление шаговыми двигателями, считывание данных с цифровых вольтметров), а также обеспечивает систему настройки работы драйверов обоих уровней. При этом результаты измерений, поступающие в программу, являются «сырыми», т.е. не анализируются и не обрабатываются.

В дальнейшем планируется создание системы, которая бы позволила подключать к клиентскому приложению модули, ответственные за конкретные экспериментальные задачи, а также создание непосредственно данных программных модулей. Такие динамически подключаемые модули смогут выполнять специфические действия по управлению устройством, анализ, преобразование и обработку поступающих данных.

Литература: 1. *Галуза А.А., Галуза А.И., Кудленко А.Д. и др.* Универсальная система автоматизации эксперимента // Радиоэлектроника и информатика. 2004. №1. С. 66-69. **2.** *Галуза А.А., Кудленко А.Д., Слатин К.А. и др.* Система автоматизации криогенного спектрального эллипсометра // Приборы и техника эксперимента. 2003. №4. С. 98-101. **3.** *Солдатов В.П.* Программирование драйверов Windows. М.: ООО «Бином-Пресс». 2004. 480 с. **4.** *Сорокина С.И. и др.* Программирование драйверов и систем безопасности. СПб.: БХВ-Петербург, М.: Издатель Молгачева С.В. 2003. 256 с. **5.** *Джеффри Рихтер.* Windows для профессионалов: Создание эффективных WIN32-приложений с учетом специфики 64-разрядной версии Windows. СПб.: Питер: М.: Издательство торговый дом «Русская редакция». 2001. 752 с. **6.** *Новиков Ю.В., Калашников О.А., Гуляев С.Э.* Разработка устройств сопряжения для персональных компьютеров типа IBM PC. М.: ЭКОМ. 1997. 160 с. **7.** *Андре Ламот.* Программирование игр для Windows. Советы профессионала. М.: Издательский дом «Вильямс». 2003. 880 с.

Поступила в редколлегию 12.11.2006

Рецензент: д-р физ.-мат. наук, проф. Мамалуй А.А.

Власенко Андрей Сергеевич, студент кафедры ПОЭВМ ХНУРЭ. Научные интересы: программирование, системное программирование. Адрес: Украина, 61100, Харьков, ул. Матюшенко, 9, кв. 34, тел. 704-27-28. E-mail: Andrew_vs@inbox.ru

Галуза Алексей Анатольевич, канд. физ.-мат. наук, доцент кафедры ПОЭВМ ХНУРЭ. Научные интересы: оптика, математическое и компьютерное моделирование физических процессов, автоматизация физического эксперимента. Адрес: Украина, 61726, Харьков, пр. Ленина, 14, тел. 702-14-46. E-mail: Galuza76@mail.ru

Захарченко Андрей Яковлевич, студент кафедры ПОЭВМ ХНУРЭ. Научные интересы: программирование, системное программирование. Адрес: Украина, 61032, Харьков, пр. Московский, 302, кв. 154, тел. 779-06-57. E-mail: Skif_poas@rambler.ru