РАЗРАБОТКА КРОССПЛАТФОРМЕННОГО ДРАЙВЕРА SERCOS-ИНТЕРФЕЙСА

Магистрант группы М-9-15 А.И. Бондаренко
Научный руководитель: д.т.н., профессор Г.М. Мартинов

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «СТАНКИН»

Работа выполнена по Госконтракту №02.740.11.0488 на проведение НИР в рамках ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009 – 2013 годы.

SERCOS (слово-акроним для словосочетания Serial Realtime Communication System) является шиной управления движением, соединяющей устройство управления, двигатели, модули ввода/вывода и датчики. SERCOS-интерфейс – единственный открытый цифровой интерфейс, являющийся международным стандартом для систем управления движением и модулями ввода/вывода, с исполнением, требуемым для синхронизации высокопроизводительных многокоординатных систем управления движением на языке, не зависимом от какого-то конкретного производителя [1].

Решение проблемы разработки и отладки систем управления движением на базе SERCOS-интерфейса возможно при использовании программного драйвера интерфейса. Основной идеей проекта было создание программного кода драйвера, допускающего произвольную обработку больших объемов данных (как внутренних SERCOS-данных, так и пользовательских) в реальном времени. Весьма перспективным представляется создание кроссплатформенного кода, компилируемого на различных платформах, (например Win32, Linux и т.д.). На текущий момент времени подобный код недоступен, может быть написан фирмами-разработчиками SERCOS-устройств для отдельных конкретных применений.

Драйвер отвечает за создание единой коммуникационной и информационной среды между операционной системой и, в конечном счете, цифровым приводом. Для реализации этого необходимо решить следующие задачи:

- коммуникационная (подразумевается совокупность программно-аппаратных средств для создания связей в системе);
- задача инициализации (под инициализацией понимается обнаружение наличия SERCOS-кольца, устройств в нем, инициализация переменных и т.д.);
- задача формирования команд управления;
- задача мониторинга параметров.

Взаимодействие между программным драйвером и контроллером привода осуществляется при помощи PCI-карты SERCANS, устанавливаемой в стандартный разъем персонального компьютера, организующей интерфейс обращения к памяти, который позволяет управлять всеми функциями, поддерживаемыми конкретным типом карты. Таковыми функциями могут быть передача циклических данных и текущих значений в реальном времени; обработка параметров привода или SERCANS-карты; прием диагностических сообщений (как от привода, так и от PCI-карты).

Предлагаемая архитектура системы представлена на рис.1.

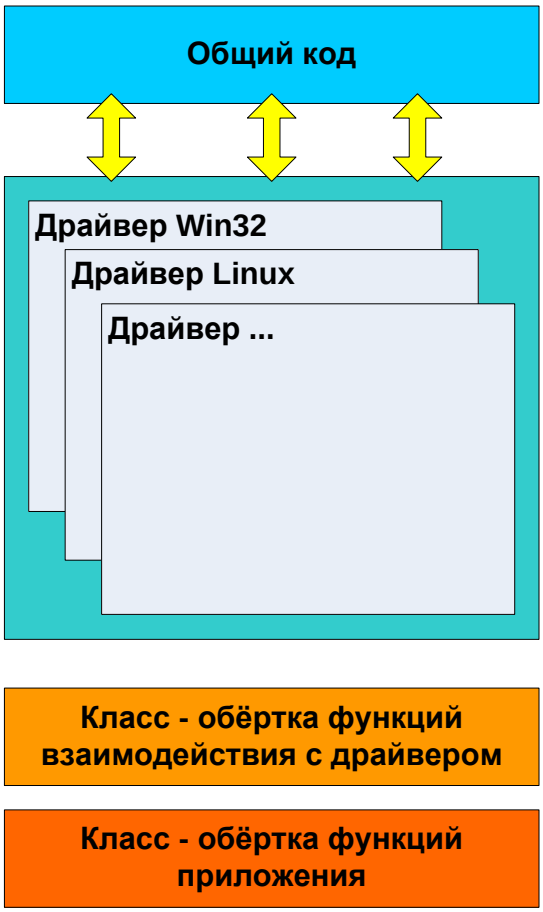

Рис.1 Основные элементы рассматриваемого решения

Выделяются следующие ключевые блоки:

- общий код;
- набор драйверов под различные платформы;
- класс-обертка функций взаимодействия с драйвером;
- класс-обертка функций приложения.

Общий код содержит в себе универсальные алгоритмы, написанные без использования функций API конкретной операционной системы, т.е. основываясь исключительно на стандартной библиотеке C-Runtime. В этом случае драйвера будут состоять только из специфичных для каждой

платформы функций (поиск устройства, нахождение базового адреса, отображение памяти и т.д.). Такая подход позволяет все остальные функции (т.е. любые действия с данными в процедуре обработки прерываний, математические вычисления, реализацию произвольных регуляторов и коррекций параметров привода), а также основные блоки данных (например, структуры для работы с SERCOS-устройствами) содержать в общем коде, который пишется на языке C/C++.

Класс-обертка функций взаимодействия с драйвером реализует специфичное поведение под ряд различных платформ. Одним из методов, при помощи которых достигается возможность компиляции приложений под разными платформами, является использование условной компиляции; следует отметить, что годится любой способ достижения мультиплатформенности программного кода. Он может включать в себя описание логики работы таких функций, как поиск устройств PCI, подписка на прерывания, выполнение команд для работы драйвера (прочитать, записать) и т.д. Например, поиск устройства SERCANS выглядит следующим образом: после загрузки драйвера сканируется шина PCI, по известным параметрам VendorID и DeviceID SERCANS-карты определяются значения базового адреса ввода-вывода и номера прерывания, которые в дальнейшем можно использовать в качестве параметров для функций из общего кода, таких как Write/ReadCycleData.

Опишем схему работы драйвера (рис.2). «Сердцем» драйвера является процедура обработки прерываний (Interrupt Service Routine, ISR). Для создания этой процедуры используется функция-обертка AttachInterrupt, позволяющая скоординировать работу потока процедуры прерывания (Interrupt Service Thread, IST) и самой процедуры прерывания (ISR), являющейся также функцией общего кода.

Непосредственно код процедуры обработки прерывания довольно-таки прост: всякий раз проверяется причина возникновения пришедшего прерывания. Если причиной прерывания является приход так называемой AT-телеграммы (Axis Telegram, AT) [2], которая передает ведущему узлу данные своего привода, тогда последовательно вызывается функции чтения циклических данных и записи информации в циклические данные. Прерывания, вызванные другими причинами, не рассматриваются. Функция чтения циклических данных ReadCycleData() считывает данные, полученные из ведомых приводов, в специальную структуру DRIVE_DATA_T. Функция записи циклических данных WriteCycleData() записывает заданные значения команд в ведущие привода. Обработка значений происходит в функции HandleCycleData(). При помощи трех этих функции организуется комплексное взаимодействие в цикле обработки прерываний.

Рис.2 Блок-схема работы драйвера

В схеме на рис.2 процедура обработки прерываний занимает верхнюю часть. В ней выделены следующие функциональные блоки – начало функции обработки прерывания, чтение текущих данных и расчет новых, фиксация данных в памяти и конец функции обработки прерывания. Чтение текущих данных и расчет новых подразумевает как обработку оперативных данных, так и конфигурационных, к которым можно отнести состояние привода (вкл./выкл., способ формирования сигнала и другие). Под фиксацией данных подразумевается реализации функции осциллографирования с дальнейшей возможностью просмотра и графического отображения записанных данных [3].

Библиографический список

1. IEC 61491, EN 61491 Sercos interface Technical short description.
2. Сосонкин В. Л., Мартинов Г. М. Архитектоника цифровых следящих приводов подач технологических машин // Мехатроника, автоматизация, управление. 2005. №10. С. 24-30.
3. Мартинов Г.М., Бондаренко А.И. Использование SERCOS-интерфейса для управления двигателями в компьютерных системах управления // VI Региональная конференция научно-практическая конференция студентов и аспирантов: сборник трудов. – Старый Оскол: СТИ НИТУ МИСИС, 2010. - С.63-66