С.А. Набоков, И.С. Соколова

МОДИФИКАЦИЯ ДРАЙВЕРА *COM*-ПОРТА (для *WINDOWS*)

Филиал «Протвино» университета «Дубна»
Кафедра информационных технологий

Рассмотрена проблема реализации снятия и установки сигнала «Запрос передачи» (RTS) на уровне ядра. Особое внимание уделено рассмотрению технологии наращивания функциональности драйвера com-порта.

Работа с устройством происходит по полудуплексному каналу (рис.1). После посылки запроса компьютер устанавливает *RTS*-сигнал (*Request To Send* - запрос передачи) на приём данных от устройства МТК-32, а по окончанию приёма данных сбрасывает этот сигнал. Данную задачу можно реализовать на уровне пользователя, но это может привести к потере данных из-за короткого интервала (1 бит) между передачей пакетов, так как установка и сброс *RTS*-сигнала происходят не стабильно во времени в зависимости от загруженности системы. Передача данных происходит по протоколу МЭК – 101, вследствие чего поставленная задача была реализована на уровне ядра.

Были поставлены следующие задачи:

- установка *RTS* сигнала перед началом передачи данных на устройство;
- снятие *RTS* сигнала по окончанию передачи данных.

Компьютер использует *RTS/CTS*-сигналы, выводы *RTS* и *CTS* (*Clear To Send* - готов к передаче) на последовательном разъеме (*RS-232*).

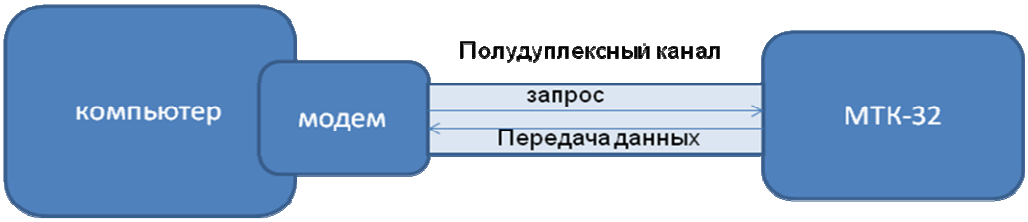

Рис.1 Общий принцип работы

Появление положительного напряжения на выводах приёмника означает сохранение посланных к нему данных. Если *RTS* инвертирован (напряжение отрицательное), то "запрос передачи данных" обратный, то есть происходит прекращение посылки данных. Когда приемник готов снова принимать данные, он устанавливает сигнал *RTS* для передатчика для продолжения передачи. Для компьютеров и терминалов вывод *RTS* посылает сигнал управления потоком данных, а вывод *CTS* получает сигнал. То есть вывод *RTS* на одном конце кабеля соединен с выводом *CTS* на другом конце.

При работе с модемом управление *RTS/CTS*-сигналами осуществляется другим способом, то есть модемный вывод *RTS* получает сигнал, а вывод *CTS* - посылает.

Наращивание функционала драйвера реализовано с помощью кодов действий. Для того чтобы установить и снять *RTS*-сигнал создаётся два кода действий. Код действий состоит из 32-х разрядного числа (рис. 2):

- “Файловый флаг (*common*)” устанавливается в случаях, когда пользователь создает новые коды действия для файловых устройств;
- “Тип устройства”: *FILE_DEVICE_CD_ROM, FILE_DEVICE_MOUSE…* - константа, характеризующая тип устройства;
- “Доступ”: *FILE_ANY_ACCESS, FILE_READ_DATA, FILE_WRITE_DATA, (FILE_READ_DATA | FILE_WRITE_DATA)*;
- “Функциональный код”: произвольное значение в диапазоне *0x800...OxFFF* (значения *0x000...0x7FF* зарезервированы для кодов *Microsoft*);
- “Тип передачи”: *METHOD_BUFFERED, METHOD_IN_DIRECT, METHOD_OUT_DIRECT, METHOD_NEITHER.*

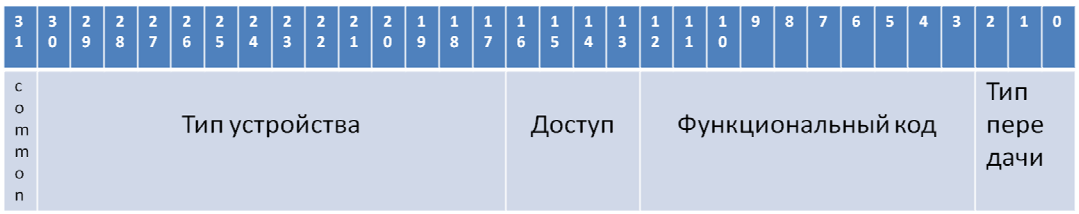

Рис.2 Структура кода действий

METHOD_BUFFERED – буферизированный ввод-вывод. Диспетчер выделяет в не подкачиваемом пуле буфер, размер которого равен наибольшему размеру, указанному в параметрах *nInBufferSize* и *nOutBufferSize* функции *DeviceIoControl*. В этот буфер копируются данные из пользовательского входного буфера (параметр *lpInBuffer*). Адрес этого буфера передается обработчику *IRP_MJ_DEVICE_CONTROL* в поле *AssociatedIrp.SystemBuffer* структуры *IRP*, а его размер – в поле *Parameters.DeviceIoControl.InputBufferLength* структуры *IO_STACK_LOCATION*. После того как обработчик драйвера был вызван, диспетчер ввода-вывода копирует возвращаемые драйвером в этом же системном буфере данные в пользовательский буфер. Размер копируемых данных *IRP*-обработчик должен указать сам, в параметре *IoStatus.Information* структуры *IRP*.

METHOD_IN_DIRECT и METHOD_OUT_DIRECT – ситуация с входным буфером аналогична буферизированному вводу-выводу. Выходной пользовательский буфер обрабатывается несколько иначе: описывающий его *MDL* помещается в поле *MdlAddress* структуры *IRP*. Входной буфер, несмотря на своё название, может служить как источником, так и приемником данных.

METHOD_NEITHER – операции по обработке как входных, так и выходных буферов целиком и полностью ложатся на плечи драйвера. В поле *DeviceIoControl.Type3InputBuffer* структуры *IO_STACK_LOCATION* содержится указатель на пользовательский входной буфер, а в поле *UserBuffer* структуры *IRP* – указатель на пользовательский выходной буфер

Коды действий вызываются из функции диспетчеризации *CtlDispatch()*.

Рассмотрим пример кода действия:

#define IOCTL_SERIAL_SET_RTS CTL_CODE (FILE_DEVICE_SERIAL_PORT, 12, METHOD_BUFFERED, FILE_ANY_ACCESS)

При работе были использованы языки программирования *C* и *C++* и среда разработки драйверов *Microsoft DKK 2003*, а также операционная система *WINDOWS XP*.

В данной работе был наращен функционал драйвера для решения поставленных задач, что позволило использовать устройства МТК-32, которые заменили аналоги старых менее функциональных устройств.

Библиографический список

1. Комиссарова, В. Программирование драйверов для WINDOWS / В. Комиссарова. – СПб. : БХВ-Петербург, 2007. - 256 с.
2. Солдатов, В.П. Программирование драйверов WINDOWS, 2-ое издание / В.П. Солдатов. СПб. : Бином-Пресс, 2004. – 432 с.
3. Baker, Art. The Windows 2000 Device Driver Book. A Guide for Programmers, Second Edition / Art Baker, Jerry Lozano. - Prentice Hall PTR : Pearson Education, 1999. – 480 p.
4. http://www.pcports.ru/ - статьи “Программирование драйверов ”.
5. http://www.linux.org.ru/books/HOWTO/Text-Terminal-HOWTO-10.html - 10.5 Аппаратное управление потоком данных (RTS/CTS и т.д.).