С. А. Набоков, В. И. Ухов

РАЗРАБОТКА ДРАЙВЕРА *COM*-ПОРТА В СРЕДЕ *WINDOWS* ДЛЯ ОБМЕНА ДАННЫМИ С МТК32 ПО ПОЛУДУПЛЕКСНОМУ КАНАЛУ СВЯЗИ

Филиал «Протвино» университета «Дубна»
Кафедра информационных технологий

Рассмотрена проблема реализации приема/передачи данных от телемеханического пункта управления МТК-30.ПУ по полудуплексному каналу связи RS-232 с устройством телемеханики контролируемого пункта МТК-32.КП

Согласно рис. 1, работа с МТК-32.КП происходит по полудуплексному каналу. После посылки запроса МТК-30.ПУ устанавливает *RTS* сигнал на приём данных от устройства МТК-32, а по окончанию приёма сбрасывает сигнал. Данную задачу можно реализовать на уровне 3-го кольца операционной системы, но это может привести к потере данных из-за короткого интервала (1 бит) между приемом/передачей пакетов, так как установка и сброс *RTS*-сигнала происходят не стабильно по времени в зависимости от загруженности системы. Передача данных происходит по протоколу МЭК–101, вследствие чего поставленная задача была реализована на уровне ядра.

RTS (Request To Send) — "Запрос передачи". Компьютер использует *RTS/CTS*-сигналы и выводы *RTS* и *CTS* на последовательном разъеме (*RS*-232).

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* (Готов к передаче ─ *Clear To Send*) получает сигнал. То есть вывод *RTS* на одном конце кабеля соединен с выводом *CTS* на другом конце.

Рис. 1. Общий принцип приёма/передачи данных

Работа коммуникационных портов реализована на универсальных асинхронных приемопередатчиках *UART*. *UART* — это микросхемы, которые работают по стандарту *RS*-232C. Для *COM*-порта компьютера используется 9-ти штырьковый разъем *DE*9p согласно стандарту *TIA*-574. В этом разъеме используется шесть сервисных сигналов и два канала обмена последовательными данными.

Полнодуплексный обмен данными означает, что можно одновременно передавать и принимать поток данных. Существуют два аппаратно и программно независимых канала передачи данных. Один канал для передачи данных, другой канал для приема данных. Причем *COM*-портам

безразлично, чем занят процессор в это время, у них присутствуют собственные буферы приема и передачи данных. В этих буферах данные выстраиваться в очередь на передачу и очередь на прочтение данных процессором. Любая программа может обратиться к *COM*-порту и получить данные из его буфера, тем самым очистив его. Естественно буферы не безграничны, их размер задается при конфигурировании портов. Интерфейсы *RS*-485, *Modbus, USB* и др. (за исключением сетевых протоколов) являются полудуплексными и физически не способны вести обмен данными в обоих направлениях одновременно.

Сервисные сигналы, предусмотренные стандартом *RS*-232c, позволяют организовать обмен данными между двумя устройствами одновременно в обоих направлениях. Сервисные сигналы представлены отдельными цифровыми входами и выходами с памятью. Например, когда по телефону на модем поступал звонок со станции, модем по 9-му контакту (*RI*) сообщал ПК, что ему позвонили, и начиналась процедура обмена данных. Причем с помощью сервисных сигналов ПК и модем могли приостановить обмен данных или заставить повторить их. Вариантов использования сервисных сигналов большое множество. Разработчик может использовать их по своему усмотрению. Например, с помощью этих сигналов удобно опрашивать контакты концевых выключателей или фотодатчиков, а также можно включать/выключать различные устройства или запитывать слаботочное устройство.

UART полностью реализован аппаратно и не зависит от программного обеспечения и ОС.

Асинхронная передача данных по каналу связи означает, что ПК может послать данные на конечное устройство, не заботясь о синхронности их поступления. Конечное устройство само подстраивается под полученные данные. В синхронных протоколах для этого служит специальный сигнал, передающийся по отдельному проводу. В коммуникационных портах синхросигнал встроен в каждый передаваемый символ в виде стартового и стопового бита. Метод, которым синхронизируются данные по стандарту *RS*-232*C*, стал общеупотребительным для всех асинхронных протоколов обмена данными.

За основу был взят стандартный драйвер *COM*-порта из пакета шаблонов *DDK*. Этот драйвер последовательно порта, который совместим с технологией *Plug-and-Play*, поддерживает динамическое подключение и удаления устройств. Код драйвера служит образцом для большинства подобных устройств.

В исходном коде драйвера было изменено два модуля — модуль отправки данных на устройство и модуль обработки прерываний. В модуль отправки была добавлена функция установки сигнала *RTS* перед началом формирования пакета данных.

```
NTSTATUS
SerialWrite(
   IN PDEVICE_OBJECT DeviceObject,
   IN PIRP Irp)
{
        ...
     SerialSetRTS(Extension); //функция для установки сигнала RTS в сигнальное состояние
        …
              return STATUS_SUCCESS;
   }
```

В модуле обработки прерываний была добавлена функции снятия сигнального состояния сигнала *RTS*. Перед снятием производится проверка 5-го бита ***THRE*** регистра состояния линии связи *LSR*. Если буфер *FIFO* пуст, о чем и сообщает данный бит, то следует полагать, что ушёл последний бит данных, отправленных на устройство. После проверки производится задержка длительностью в 3 бита. Она служит гарантией того, что не произойдет наложение данных между отправками пакетов данных.

```
    while (1) //проверка 5-го бита THRE регистра состояния линии связи LSR
{
registr = (READ_LINE_STATUS(Extension->Controller) & (SERIAL_LSR_TEMT));
if (registr!=0) break;
}

    KeStallExecutionProcessor(timeout); // установка задержки
```

В результате внесенных в структуру стандартного драйвера *COM*-порта изменений удалось добиться того, что сигнал *RTS* снимается не более чем через 1 бит, а это указывает на высокую точность переключения сигнала, что удовлетворяет требованиям поставленной задачи.

При работе были использованы языки программирования *C* и *C++* и среда разработки драйверов *Microsoft DKK* 2003, а также операционная система *WINDOWS XP SP3.*

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* БХВ-Петербург, 2007. — 256 с.

2. Солдатов В .П. "Программирование драйверов *WINDOWS*", 2-ое издание / *В. П. Солдатов* — Бином-Пресс, 2004. — 576 с.

3. http://www.pcports.ru/ (статьи "Программирование драйверов")

4. http://www.linux.org.ru/books/HOWTO/Text-Terminal-HOWTO-10.html ("10.5 Аппаратное управление потоком данных *(RTS/CTS* и т. д.)")