

ПАВЕЛ АГУРОВ



ПРАКТИКА ПРОГРАММИРОВАНИЯ USB



USB 1.1/2.0

КЛАССЫ HID, CDC
И ПРИМЕРЫ ИХ РЕАЛИЗАЦИИ

ФУНКЦИИ РАБОТЫ С USB
ДЛЯ WINDOWS 98/NT/2000/XP

РАЗРАБОТКА USB-ДРАЙВЕРОВ
ДЛЯ WINDOWS 2000/XP

ПРИМЕРЫ ПРОГРАММ
НА ЯЗЫКАХ DELPHI, C, C#

СПЕЦИАЛЬНЫЕ ФУНКЦИИ
RAW INPUT, DIRECT INPUT,
SETUP API

PRO

ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ



Павел Агуров

**ПРАКТИКА
ПРОГРАММИРОВАНИЯ
USB**

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.1
А27

Агуров П. В.

А27 Практика программирования USB. — СПб.: БХВ-Петербург, 2006. — 624 с.: ил.

ISBN 978-5-94157-851-1

В книге собрана информация, необходимая для создания USB-устройств и драйверов для операционной системы Microsoft Windows 2000/XP. Рассмотрен процесс создания USB-устройства: от написания программы микроконтроллера (примеры реализованы для микропроцессора AT89C5131) до разработки собственного WDM-драйвера. Содержится описание специальных классов устройств: HID-класс, позволяющий обойтись без разработки драйвера, и класс CDC, позволяющий работать с USB как с обычным COM-портом. Рассмотрено использование функций Raw Input, Direct Input и Setup API, содержится большое количество практических советов и примеров программ на языках Delphi, C и C#. Для удобства читателей все исходные коды приводятся на прилагаемом компакт-диске.

Для программистов и разработчиков аппаратуры

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Константин Костенко</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Наталья Першакова</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.03.06.
Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 50,31.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-94157-851-1

© Агуров П. В., 2006
© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Введение	1
Для кого эта книга.....	1
Что вы найдете в книге.....	2
Программные требования.....	2
Аппаратные требования	2
О программном коде	3
Краткое описание глав.....	3
Обозначения.....	5
Благодарности	6
Обратная связь	6
Часть I. ОБЩИЕ СВЕДЕНИЯ О USB	7
Глава 1. Спецификация USB.....	9
1.1. Что такое USB и зачем это надо.....	9
1.1.1. Общая архитектура USB	11
1.1.2. Физическая и логическая архитектура USB	11
1.1.3. Составляющие USB.....	13
1.1.4. Свойства USB-устройств.....	14
1.1.5. Принципы передачи данных	15
1.1.6. Механизм прерываний	15
1.1.7. Режимы передачи данных.....	15
1.1.8. Логические уровни обмена данными	16
1.1.8.1. Уровень клиентского ПО.....	17
1.1.8.2. Уровень системного драйвера USB	17
1.1.8.3. Уровень хост-контроллера интерфейса.....	19
1.1.8.4. Уровень шины периферийного USB-устройства	19
1.1.8.5. Уровень логического USB-устройства	19
1.1.8.6. Функциональный уровень USB-устройства	19
1.1.9. Передача данных по уровням.....	20
1.1.10. Типы передач данных.....	21
1.1.11. Кадры	24
1.1.12. Конечные точки.....	25
1.1.13. Каналы	26
1.1.14. Пакеты.....	27
1.1.14.1. Формат маркер-пакетов IN, OUT, SETUP и PING	29

1.1.14.2. Формат пакета SOF	30
1.1.14.3. Формат пакета данных	30
1.1.14.4. Формат пакета подтверждения	31
1.1.14.5. Формат пакета SPLIT	31
1.1.15. Контрольная сумма	32
1.1.15.1. Алгоритм вычисления CRC	32
1.1.15.2. Программное вычисление CRC	34
1.1.16. Транзакции	37
1.1.16.1. Типы транзакций	37
1.1.16.2. Подтверждение транзакций и управление потоком	38
1.1.16.3. Протоколы транзакций	40
1.2. Запросы к USB-устройствам	42
1.2.1. Конфигурационный пакет	42
1.2.2. Стандартные запросы к USB-устройствам	46
1.2.2.1. Получение состояния <i>GET_STATUS</i>	46
1.2.2.2. Сброс свойства <i>CLEAR_FEATURE</i>	47
1.2.2.3. Разрешение свойства <i>SET_FEATURE</i>	47
1.2.2.4. Задание адреса на шине <i>SET_ADDRESS</i>	48
1.2.2.5. Получение дескриптора <i>GET_DESCRIPTOR</i>	48
1.2.2.6. Передача дескриптора <i>SET_DESCRIPTOR</i>	49
1.2.2.7. Получение кода конфигурации <i>GET_CONFIGURATION</i>	49
1.2.2.8. Задание кода конфигурации <i>SET_CONFIGURATION</i>	50
1.2.2.9. Получение кода настройки интерфейса <i>GET_INTERFACE</i>	50
1.2.2.10. Задание кода настройки интерфейса <i>SET_INTERFACE</i>	50
1.2.2.11. Задание номера кадра синхронизации <i>SYNC_FRAME</i>	51
1.2.2.12. Обработка стандартных запросов	51
1.2.3. Дескриптор устройства	52
1.2.3.1. Дескриптор устройства	52
1.2.3.2. Уточняющий дескриптор устройства	55
1.2.3.3. Дескриптор конфигурации	56
1.2.3.4. Дескриптор интерфейса	59
1.2.3.5. Дескриптор конечной точки	60
1.2.3.6. Дескриптор строки	63
1.2.3.7. Специфические дескрипторы	64
1.2.3.8. Порядок получения дескрипторов	65
1.3. Система Plug and Play (PnP)	69
1.3.1. Конфигурирование USB-устройств	70
1.3.2. Нумерация USB-устройств	70
1.3.3. PnP-идентификаторы USB-устройств	72
1.3.4. Символьные имена устройств	73
1.4. Модель WDM	74

Глава 2. Программирование на языке C для микроконтроллера 78

2.1. Общие сведения о языке C для микроконтроллеров	78
2.2. Использование стандартных библиотек	80

2.3. Программирование для AT89C5131	80
2.3.1. Файл инициализации	81
2.3.2. Структуры дескрипторов.....	83
2.3.3. Структура проекта	85
Глава 3. Инструменты	87
3.1. Программаторы	87
3.1.1. Программатор Flip.....	87
3.1.2. Программатор ER-Tronik.....	91
3.2. Инструменты создания драйверов	94
3.2.1. NuMega Driver Studio	94
3.2.2. Jungo WinDriver.....	94
3.2.3. Jungo KernelDriver.....	94
3.3. Средства Microsoft Visual Studio	94
3.3.1. Depends (Dependency Walker)	95
3.3.2. Error Lookup	95
3.3.3. GuidGen	95
3.4. Средства Microsoft DDK	96
3.4.1. DeviceTree	97
3.4.2. DevCon.....	97
3.4.2.1. Ключ <i>classes</i>	98
3.4.2.2. Ключ <i>driverfiles</i>	98
3.4.2.3. Ключ <i>hwids</i>	100
3.4.2.4. Ключ <i>rescan</i>	101
3.4.2.5. Ключ <i>stack</i>	101
3.4.2.6. Ключ <i>status</i>	102
3.4.3. ChkInf и GenInf.....	103
3.5. Средства CompuWare Corporation	103
3.5.1. Monitor	103
3.5.2. SymLink.....	104
3.5.3. EzDriverInstaller	104
3.5.4. WdmSniff	105
3.6. Средства SysInternals.....	106
3.6.1. WinObj.....	106
3.7. Средства USB Forum	106
3.7.1. HID Descriptor Tool.....	106
3.8. USB Command Verifier.....	109
3.9. Средства HDD Software.....	111
3.10. Средства Sourceforge	111
3.11. Программа мониторинга Bus Hound	112
Глава 4. Принципы использования функций Win32 в .NET	114
4.1. Общие сведения.....	114
4.2. Импорт функций Win32.....	115

4.3. Структуры	116
4.3.1. Атрибут <i>StructLayout</i>	117
4.3.2. Атрибут <i>MarshalAs</i>	118
4.4. Прямой доступ к данным	120
4.5. Обработка сообщений Windows	121
4.6. Общие сведения о WMI	122
4.7. Интернет-ресурсы к этой главе.....	124

ЧАСТЬ II. КЛАССЫ USB..... 125

Глава 5. Класс CDC 127

5.1. Методы преобразования интерфейсов USB/RS-232.....	127
5.2. Общие сведения об интерфейсе RS-232.....	128
5.2.1. Линии обмена.....	129
5.2.1.1. Передаваемые данные (BA/TxD/TD)	130
5.2.1.2. Принимаемые данные (BB/RxD/RD)	130
5.2.1.3. Запрос передачи (CA/RTS).....	130
5.2.1.4. Готовность к передаче (CB/CTS).....	131
5.2.1.5. Готовность DCE (CC/DSR).....	132
5.2.1.6. Готовность DTE (CD/DTR).....	132
5.2.1.7. Индикатор вызова (CE/RI).....	132
5.2.1.8. Обнаружение несущей (CF/DCD).....	132
5.2.1.9. Готовность к приему (CJ)	133
5.3. Спецификация CDC.....	133
5.3.1. Стандартные дескрипторы.....	133
5.3.2. Функциональные дескрипторы.....	135
5.3.2.1. Заголовочный функциональный дескриптор	136
5.3.2.2. Дескриптор режима команд	137
5.3.2.3. Дескриптор абстрактного устройства	138
5.3.2.4. Дескриптор группирования	138
5.3.3. Специальные запросы	139
5.3.3.1. Запрос <i>SET_LINE_CODING</i>	139
5.3.3.2. Запрос <i>GET_LINE_CODING</i>	140
5.3.3.3. Запрос <i>SET_CONTROL_LINE_STATE</i>	140
5.3.3.4. Запрос <i>SEND_BREAK</i>	141
5.3.4. Нотификации	141
5.3.4.1. Нотификация <i>RING_DETECT</i>	141
5.3.4.2. Нотификация <i>SERIAL_STATE</i>	141
5.4. Поддержка CDC в Windows.....	142
5.4.1. Обзор функций Windows для работы с последовательными портами.....	142
5.4.1.1. Основные операции с портом.....	142
5.4.1.2. Функции настройки порта.....	143
5.4.1.3. Специальная настройка порта.....	143

5.4.1.4. Получение состояния линий модема	144
5.4.1.5. Работа с CDC на платформе .NET	144
5.4.2. Соответствие функций Windows и USB-запросов	144
Глава 6. Класс HID.....	146
6.1. Спецификация HID-устройств	146
6.2. Порядок обмена данными с HID-устройством.....	148
6.3. Установка драйвера HID-устройства.....	149
6.4. Идентификация HID-устройства.....	149
6.4.1. Идентификация загрузочных устройств.....	150
6.4.2. Дескриптор конфигурации HID-устройства	150
6.4.3. HID-дескриптор.....	151
6.4.4. Дескриптор репорта.....	153
6.5. Структура дескриптора репорта	153
6.5.1. Элементы репорта.....	154
6.5.1.1. Элементы короткого типа.....	154
6.5.1.2. Элементы длинного типа.....	154
6.5.2. Типы элементов репорта.....	155
6.5.2.1. Основные элементы	155
6.5.2.2. Глобальные элементы.....	158
6.5.2.3. Локальные элементы	161
6.5.3. Примеры дескрипторов.....	162
6.6. Запросы к HID-устройству.....	165
6.6.1. Запрос <i>GET_REPORT</i>	166
6.6.2. Запрос <i>SET_REPORT</i>	167
6.6.3. Запрос <i>GET_IDLE</i>	167
6.6.4. Запрос <i>SET_IDLE</i>	168
6.6.5. Запрос <i>GET_PROTOCOL</i>	168
6.6.6. Запрос <i>SET_PROTOCOL</i>	169
6.7. Инструменты	169
6.8. Драйверы для HID-устройств в Windows.....	169
Глава 7. Другие классы USB.....	179
Часть III. ПРАКТИКА ПРОГРАММИРОВАНИЯ USB.....	181
Глава 8. Создание USB-устройства на основе AT89C5131.....	183
8.1. Общая информация об AT89C5131.....	183
8.2. Структурная схема AT89C5131	185
8.3. USB-регистры AT89C5131.....	187
8.3.1. Регистр <i>USBCON</i>	187
8.3.2. Регистр <i>USBADDR</i>	189
8.3.3. Регистр <i>USBINT</i>	190
8.3.4. Регистр <i>USBIEN</i>	191

8.3.5. Регистр <i>UEPNUM</i>	192
8.3.6. Регистр <i>UEPCONX</i>	193
8.3.7. Регистр <i>UEPSTAX</i>	195
8.3.8. Регистр <i>UEPRST</i>	197
8.3.9. Регистр <i>UEPINT</i>	198
8.3.10. Регистр <i>UEPIEN</i>	199
8.3.11. Регистр <i>UEPDATX</i>	200
8.3.12. Регистр <i>UBYCTLX</i>	201
8.3.13. Регистр <i>UFNUML</i>	201
8.3.14. Регистр <i>UFNUMH</i>	201
8.4. Схемотехника AT89C5131	202
8.5. Базовый проект для AT89C5131	204
8.5.1. Первая версия программы для AT89C5131	204
8.5.2. Добавляем строковые дескрипторы.....	226
8.5.3. Добавление конечных точек.....	231
8.6. Загрузка программы.....	235
Глава 9. Реализация класса CDC	236
9.1. Реализация CDC	236
9.2. Дескрипторы устройства	237
9.2.1. Инициализация конечных точек	240
9.2.2. Обработка CDC-запросов.....	241
9.2.3. Конфигурирование RS-порта и CDC-линии.....	243
9.2.4. Прием и передача данных	244
9.3. Установка драйвера.....	247
9.4. Программирование обмена данными с CDC-устройством на языке Delphi	249
9.5. Программирование обмена с CDC-устройством на языке C#.....	279
9.5.1. Использование компонента MSCOMM.....	279
9.5.2. Использование функций Win32	283
9.6. Проблемы CDC.....	289
Глава 10. Реализация класса HID	290
10.1. Реализация HID на AT89C5131	290
10.2. Передача нескольких байтов	296
10.3. Feature-репорты.....	298
10.4. Передача данных от хоста (<i>SET_REPORT</i>)	300
10.5. Установка HID-устройства	301
10.6. Обмен данными с HID-устройством	301
10.6.1. Получение имени HID-устройства	302
10.6.2. Получение атрибутов устройства и чтение репортов.....	305
10.6.3. Передача данных от хоста к HID-устройству.....	311
10.7. Примеры HID-устройств	312
10.7.1. Реализация устройства "мышь"	312
10.7.2. Реализация устройства "клавиатура".....	313

10.8. Использование HID-протокола	316
10.8.1. Интерпретация данных	318
10.8.2. Коллекции	320
10.8.3. Массивы и кнопки.....	326
10.9. HID-устройство с несколькими репортами	329
Глава 11. Специальные функции Windows	332
11.1. Функции Setup API.....	332
11.1.1. Перечисление USB-устройств	332
11.1.2. Получение состояния USB-устройства	342
11.2. Перечисление USB-устройств с помощью WMI.....	345
11.3. Специальные функции Windows XP.....	347
11.3.1. <i>HidD_GetInputReport</i> — чтение HID-репортов.....	347
11.3.2. Получение данных Raw Input	348
11.4. Функции DirectX.....	357
11.5. Диалог добавления нового оборудования	363
11.6. Работа с символьными именами устройств	364
11.7. Безопасное извлечение флэш-дисков.....	368
11.8. Обнаружение добавления и удаления устройств	374
11.9. Интернет-ресурсы	378
Глава 12. Разработка драйвера.....	379
12.1. Основные процедуры драйвера WDM.....	379
12.1.1. Процедура <i>DriverEntry</i>	379
12.1.2. Процедура <i>AddDevice</i>	382
12.1.3. Процедура <i>Unload</i>	384
12.1.4. Рабочие процедуры драйвера.....	386
12.1.4.1. Заголовок пакета.....	386
12.1.4.2. Ячейки стека ввода/вывода	387
12.1.4.3. Рабочие процедуры драйвера	388
12.1.5. Обслуживание запросов IOCTL	393
12.2. Загрузка драйвера и обращение к процедурам драйвера.....	399
12.2.1. Процедура работы с драйвером.....	399
12.2.2. Регистрация драйвера	401
12.2.2.1. Регистрация с помощью SCM-менеджера	401
12.2.2.2. Параметры драйвера в реестре	406
12.2.3. Обращение к рабочим процедурам	408
12.2.4. Хранение драйвера внутри исполняемого файла	409
12.3. Создание драйвера с помощью Driver Studio.....	411
12.3.1. Несколько слов о библиотеке Driver Studio	413
12.3.1.1. Класс <i>KDriver</i>	413
12.3.1.2. Класс <i>KDevice</i>	413
12.3.1.3. Класс <i>KIrp</i>	414
12.3.1.4. Класс <i>KRegistryKey</i>	414
12.3.1.5. Класс <i>KLowerDevice</i>	414
12.3.1.6. Классы USB.....	416

12.3.2. Другие классы Driver Studio	417
12.3.3. Создание шаблона драйвера с помощью Driver Studio.....	417
12.3.3.1. Шаг 1. Задание имени и пути проекта.....	418
12.3.3.2. Шаг 2. Выбор архитектуры драйвера.....	418
12.3.3.3. Шаг 3. Выбор шины	418
12.3.3.4. Шаг 4. Задание набора конечных точек.....	420
12.3.3.5. Шаг 5. Задание имени класса и файла.....	421
12.3.3.6. Шаг 6. Выбор функций драйвера.....	421
12.3.3.7. Шаг 7. Выбор способа обработки запросов.....	423
12.3.3.8. Шаг 8. Создание сохраняемых параметров драйвера	424
12.3.3.9. Шаг 9. Свойства драйвера.....	425
12.3.3.10. Шаг 10. Задание кодов IOCTL	427
12.3.3.11. Шаг 11. Дополнительные настройки.....	427
12.3.4. Доработка шаблона драйвера	429
12.3.5. Базовые методы класса устройства	429
12.3.6. Реализация чтения данных	433
12.3.7. Установка драйвера.....	434
12.3.8. Программа чтения данных.....	435
12.3.9. Чтение данных с конечных точек других типов.....	445
12.3.10. "Чистый" драйвер USB-устройства	446
ЧАСТЬ IV. СПРАВОЧНИК.....	465
Глава 13. Формат INF-файла.....	467
13.1. Структура INF-файла	467
13.1.1. Секция <i>Version</i>	468
13.1.2. Секция <i>Manufacturer</i>	470
13.1.3. Секция <i>DestinationDirs</i>	472
13.1.3.1. Ключ <i>DefaultDescDir</i>	472
13.1.3.2. Ключи <i>file-list-section</i>	472
13.1.3.3. Ключ <i>dirid</i>	473
13.1.3.4. Ключ <i>subdir</i>	474
13.1.4. Секция описания модели.....	474
13.1.5. Секция <i>xxx.AddReg</i> и <i>xxx.DelReg</i>	475
13.1.6. Секция <i>xxx.LogConfig</i>	477
13.1.7. Секция <i>xxx.CopyFiles</i>	477
13.1.8. Секция <i>Strings</i>	479
13.1.9. Связи секций.....	479
13.2. Создание и тестирование INF-файлов	480
13.3. Установка устройств с помощью INF-файла.....	482
13.4. Ветки реестра для USB.....	482
Глава 14. Базовые функции Windows	484
14.1. Функции <i>CreateFile</i> и <i>CloseHandle</i> : открытие и закрытие объекта.....	484
14.1.1. Дополнительные сведения	485

14.1.2. Возвращаемое значение	486
14.1.3. Пример вызова	486
14.2. Функция <i>ReadFile</i> : чтение данных	488
14.2.1. Дополнительные сведения	489
14.2.2. Возвращаемое значение	490
14.2.3. Пример вызова	490
14.3. Функция <i>WriteFile</i> : передача данных	491
14.3.1. Дополнительные сведения	492
14.3.2. Возвращаемое значение	492
14.3.3. Пример вызова	493
14.4. Функция <i>ReadFileEx</i> : APC-чтение данных	494
14.4.1. Возвращаемое значение	495
14.4.2. Дополнительные сведения	495
14.4.3. Пример вызова	496
14.5. Функция <i>WriteFileEx</i> : APC-передача данных	496
14.5.1. Возвращаемое значение	497
14.5.2. Пример вызова	497
14.6. Функция <i>WaitForSingleObject</i> : ожидание сигнального состояния объекта	498
14.6.1. Возвращаемое значение	499
14.7. Функция <i>WaitForMultipleObjects</i> : ожидание сигнального состояния объектов	499
14.7.1. Возвращаемое значение	500
14.8. Функция <i>GetOverlappedResult</i> : результат асинхронной операции	501
14.8.1. Возвращаемое значение	502
14.9. Функция <i>DeviceIoControl</i> : прямое управление драйвером	502
14.9.1. Возвращаемое значение	504
14.10. Функция <i>CancelIo</i> : прерывание операции	504
14.10.1. Возвращаемое значение	504
14.11. Функция <i>QueryDosDevice</i> : получение имени устройства по его DOS-имени	505
14.11.1. Возвращаемое значение	505
14.11.2. Пример вызова	506
14.12. Функция <i>DefineDosDevice</i> : операции с DOS-именем устройства	507
14.12.1. Возвращаемое значение	507
14.12.2. Пример вызова	507

Глава 15. Структуры и функции Windows для последовательных портов 509

15.1. Структура настроек порта <i>COMMCONFIG</i>	509
15.2. Структура свойств порта <i>COMMPROP</i>	511
15.3. Структура тайм-аутов <i>COMMTIMEOUTS</i>	518
15.4. Структура статуса порта <i>COMSTAT</i>	520
15.5. Структура <i>DCB</i>	522
15.6. Функция <i>BuildCommDCB</i> : создание структуры <i>DCB</i> из строки	528
15.6.1. Дополнительные сведения	530

15.6.2. Возвращаемое значение	530
15.6.3. Пример вызова	530
15.7. Функция <i>BuildCommDCBAndTimeouts</i> : создание структуры <i>DCB</i> и тайм-аутов из строки	531
15.8. Функции <i>SetCommBreak</i> и <i>ClearCommBreak</i> : управление выводом данных	531
15.8.1. Возвращаемое значение	532
15.9. Функция <i>ClearCommError</i> : получение и сброс ошибок порта	532
15.9.1. Возвращаемое значение	533
15.10. Функция <i>EscapeCommFunction</i> : управление портом	534
15.10.1. Возвращаемое значение	534
15.11. Функции <i>GetCommMask</i> и <i>SetCommMask</i> : маска вызова событий	535
15.11.1. Возвращаемое значение	535
15.12. Функция <i>WaitCommEvent</i> : ожидание события СОМ-порта	536
15.12.1. Возвращаемое значение	537
15.12.2. Дополнительные сведения	537
15.12.3. Пример вызова	537
15.13. Функции <i>GetCommConfig</i> и <i>SetCommConfig</i> : конфигурирование параметров порта	540
15.13.1. Возвращаемое значение	541
15.13.2. Пример вызова	541
15.14. Функция <i>CommConfigDialog</i> : диалог конфигурирования порта	542
15.14.1. Возвращаемое значение	543
15.14.2. Дополнительные сведения	543
15.14.3. Пример вызова	543
15.15. Функция <i>GetCommProperties</i> : прочитать свойства порта	544
15.15.1. Возвращаемое значение	544
15.15.2. Пример вызова	544
15.16. Функции <i>GetCommState</i> и <i>SetCommState</i> : состояние порта	544
15.16.1. Возвращаемое значение	545
15.16.2. Пример вызова	545
15.17. Функции <i>GetCommTimeouts</i> и <i>SetCommTimeouts</i> : тайм-ауты порта	546
15.17.1. Возвращаемое значение	547
15.17.2. Пример вызова	547
15.18. Функция <i>PurgeComm</i> : сброс буферов порта	547
15.18.1. Возвращаемое значение	548
15.18.2. Пример вызова	548
15.19. Функция <i>SetupComm</i> : конфигурирование размеров буферов	548
15.19.1. Возвращаемое значение	550
15.20. Функции <i>GetDefaultCommConfig</i> и <i>SetDefaultCommConfig</i> : настройки порта по умолчанию	550
15.20.1. Возвращаемое значение	551
15.21. Функция <i>TransmitCommChar</i> : передача специальных символов	551
15.21.1. Возвращаемое значение	551

15.22. Функция <i>GetCommModemStatus</i> : статус модема	552
15.22.1. Возвращаемое значение	552
15.22.2. Пример вызова	553
15.23. Функция <i>EnumPorts</i> : перечисление портов	553
15.23.1. Дополнительные сведения	555
15.23.2. Возвращаемое значение	555
15.23.3. Пример вызова	555
Глава 16. Структуры и функции Windows Setup API	557
16.1. Функция <i>SetupDiGetClassDevs</i> : перечисление устройств	557
16.1.1. Возвращаемое значение	558
16.2. Функция <i>SetupDiDestroyDeviceInfoList</i> : освобождение блока описания устройства	558
16.2.1. Возвращаемое значение	559
16.3. Функция <i>SetupDiEnumDeviceInterfaces</i> : информация об устройстве	559
16.3.1. Возвращаемое значение	560
16.4. Функция <i>SetupDiGetDeviceInterfaceDetail</i> : детальная информация об устройстве	561
16.5. Функция <i>SetupDiEnumDeviceInfo</i> : информация об устройстве	562
16.6. Функция <i>SetupDiGetDeviceRegistryProperty</i> : получение Plug and Play свойств устройства	564
16.7. Функция <i>CM_Get_DevNode_Status</i> : статус устройства	565
16.8. Функция <i>CM_Request_Device_Eject</i> : безопасное извлечение устройства	566
Глава 17. Структуры и функции Windows HID API	568
17.1. Функция <i>HidD_Hello</i> : проверка библиотеки	568
17.2. Функция <i>HidD_GetHidGuid</i> : получение GUID	569
17.3. Функция <i>HidD_GetPreparsedData</i> : создание описателя устройства	570
17.4. Функция <i>HidD_FreePreparsedData</i> : освобождение описателя устройства	571
17.5. Функция <i>HidD_GetFeature</i> : получение Feature-репорта	571
17.6. Функция <i>HidD_SetFeature</i> : передача Feature-репорта	573
17.7. Функция <i>HidD_GetNumInputBuffers</i> : получение числа буферов	573
17.8. Функция <i>HidD_SetNumInputBuffers</i> : установка числа буферов	574
17.9. Функция <i>HidD_GetAttributes</i> : получение атрибутов устройства	575
17.10. Функция <i>HidD_GetManufacturerString</i> : получение строки производителя	576
17.11. Функция <i>HidD_GetProductString</i> : получение строки продукта	578
17.12. Функция <i>HidD_GetSerialNumberString</i> : получение строки серийного номера	578
17.13. Функция <i>HidD_GetIndexedString</i> : получение строки по индексу	579
17.14. Функция <i>HidD_GetInputReport</i> : получение Input-репорта	580
17.15. Функция <i>HidD_SetOutputReport</i> : передача Output-репорта	581

17.16. Функция <i>HidP_GetCaps</i> : получение свойств устройства	581
17.17. Функция <i>HidP_MaxDataListLength</i> : получение размеров репортов	584
17.18. Функция <i>HidD_FlushQueue</i> : сброс буферов	585
17.19. Функция <i>HidP_GetLinkCollectionNodes</i> : дерево коллекций	585
17.20. Функции <i>HidP_GetScaledUsageValue</i> и <i>HidP_SetScaledUsageValue</i> : получение и задание преобразованных значений	586
17.21. Функция <i>HidP_MaxUsageListLength</i> : размер буфера для кодов клавиш	587
17.22. Функция <i>HidP_UsageListDifference</i> : различие между массивами	587
ПРИЛОЖЕНИЯ	589
Приложение 1. Дополнительные функции	591
Приложение 2. Компиляция примеров в других версиях Delphi	592
Приложение 3. Таблица идентификаторов языков (LangID)	593
Приложение 4. Таблица кодов производителей (Vendor ID, Device ID)	596
Приложение 5. Как создать ярлык Device Manager	598
Приложение 6. Часто задаваемые вопросы	599
Приложение 7. Описание компакт-диска	600
Литература	602
Предметный указатель	603

Введение

Со дня выпуска книги "Интерфейс USB. Практика использования и программирования" прошел почти год. Судя по количеству писем, она оказалась востребована, и появилась идея второй книги, основанной на вопросах читателей и являющейся логическим продолжением первой.

В процессе дискуссий с читателями выяснились несколько основных направлений, вызывающих наибольший интерес: USB-устройства, не требующие написания драйверов (HID, CDC), подключение USB-устройств, как виртуальных последовательных портов, а также реализация всех алгоритмов на современном языке C#. В этом издании мы постараемся дать информацию об этих и многих других вопросах.

Кроме того, мы рассмотрим некоторые функции, добавленные в операционную систему Windows XP, значительно облегчающие работу с USB-интерфейсом. Так, например, функции Raw Input позволяют получать данные HID-устройств с помощью одной-двух строк кода (без использования сложных функций поиска и открытия устройства). Мы будем также обсуждать функции DirectX для чтения данных HID-устройства — класс функций `DirectInput`.

Для кого эта книга

Данная книга предназначена для программистов-практиков. Она не содержит сведений об аппаратной части USB, а полностью посвящена вопросам программирования.

Эта книга для вас, если:

- вы хотите научиться использовать современный USB-интерфейс;
- вы хотите разработать USB-устройство, работающее без установки драйверов;
- вы хотите работать с USB-устройством как с обычным последовательным портом;
- вы хотите разработать свой драйвер для USB-устройства.

Что вы найдете в книге

Эта книга посвящена USB-интерфейсу, но вы не найдете в ней сведений об электрических характеристиках и аппаратной части, зато найдете множество исходных кодов, примеров программ и просто полезных советов. Книга содержит сведения, необходимые для работы с USB-интерфейсом и написания драйверов. На прилагаемом компакт-диске содержатся полные исходные тексты и скомпилированные модули программ.

Программные требования

На компьютере должна быть установлена либо Windows 2000, либо Windows XP. Желательно при этом наличие всех доступных пакетов обновлений (Service Pack). Возможно использование Windows 98, но многие примеры, связанные с написанием драйверов, в ней работать не будут.

Для написания программ мы будем использовать пакеты Borland Delphi 6, Visual Studio 6 и Microsoft C# (.NET 1.1). Так как мы не будем использовать никаких специфических функций, присущих именно этим средствам разработки, то все примеры могут быть скомпилированы в других версиях этих пакетов, практически без модификации (см. приложение 2). Для компиляции драйверов, приведенных в книге, потребуется Windows 2000 DDK или Windows XP DDK, в соответствии с вашей версией Windows.

Как дополнительный источник информации мы рекомендуем установить MSDN¹.

Аппаратные требования

Достаточно обычного домашнего компьютера, на котором компиляция программы в Borland Delphi или Visual Studio занимает приемлемое для вас время. Установка Borland Delphi 6 потребует примерно 300 Мбайт на жестком диске, установка Visual Studio — 240 Мбайт, MSDN — 1,5 Гбайт, Windows DDK — 700 Мбайт, C# — 300 Мбайт, .NET 1.1 — 70 Мбайт. Для тестирования программ необходимо иметь одно или несколько USB-устройств. Для создания своих устройств будет необходим соответствующий инструментарий.

¹ MSDN (Microsoft Developer Network) — собрание документов компании Microsoft, содержащее сведения обо всех ее разработках.

О программном коде

Вопрос о языках программирования, используемых в книге, сложен. Мы признаемся, что так и не смогли выбрать между Delphi, C и C#. Книги [3] и [4] были написаны для языка Delphi, однако, количество писем читателей, желающих получить код для языка C, примерно одинаково с количеством писем читателей, использующих Delphi. А актуальность и нарастающая популярность C# не дают возможности не приводить примеры для платформы .NET. Идя на компромисс, мы приведем многие примеры на Delphi, примеры, специфические для .NET, будут использовать язык C#. Для языка C мы приведем заголовки основных структур и функций.

Книга содержит полные исходные коды всех программ, однако многие листинги содержат только изменения кода относительно предыдущего рассмотренного. Такое сокращение позволяет не только экономить место, но и улучшить понимание, делая акцент только на новой функциональности. Код на компакт-диске содержит полные тексты, без сокращений.

В программах на Delphi не приводится код самого проекта (DPR-файл) и код формы (DFM-файлы) — их можно найти на компакт-диске. Следует отдельно заметить (этот вопрос очень часто возникал у читателей первой книги), что для компиляции Delphi-программ, требуется либо вручную подключать необходимые функции (см. листинг 7.6), либо использовать модули из библиотеки JEDI (<http://delphi-jedi.org>). Мы специально не включали библиотеку на компакт-диск, т. к. читатели всегда могут получить последнюю версию с официального сайта. *Подробную информацию о компиляции примеров в различных версиях Delphi см. в приложении 2.*

Краткое описание глав

Главы 1—4 составляют первую часть книги. В ней приводятся общие сведения и спецификации USB, объясняются общие принципы программирования на языке C для микроконтроллера и принципы программирования аппаратуры на платформе .NET. Кроме того, первая часть содержит главу об инструментах, делающих работу с USB более простой и комфортной.

- *Глава 1* ("Спецификация USB") содержит описание USB-интерфейса: архитектуру, механизмы передачи данных, конечные точки, форматы пакетов, запросы. Кроме того, в этой главе содержатся общие сведения о системе Plug and Play и драйверной модели WDM (Windows Driver Model).
- *Глава 2* ("Программирование на языке C для микроконтроллера") содержит информацию об общих принципах программирования на языке C для микроконтроллера на примере AT89C5131. Здесь мы приводим базовый проект, которым будем пользоваться при построении приме-

ров. Программисты, знакомые с языком C, могут смело пропустить эту главу.

- *Глава 3* ("Инструменты") содержит описание программ, которые могут оказаться полезными при работе с USB-интерфейсом.
- *Глава 4* ("Принципы использования функций Win32 в .NET") содержит информацию, необходимую для программирования аппаратуры на платформе .NET.

Главы 5—7 составляют вторую часть книги, посвященную классам USB-устройств, использование которых избавляет от создания собственных драйверов.

- *Глава 5* ("Класс CDC") содержит описание класса коммуникационных устройств. Устройства, которые мы рассмотрим, "видны" операционной системе как обычный последовательный порт. В этой же главе приведено описание функций Windows для работы с такими устройствами (как для Win32, так и для .NET).
- *Глава 6* ("Класс HID") содержит описание класса устройств связи с пользователем, который позволяет создавать широкий спектр USB-устройств, не требующих написания дополнительных драйверов.
- *Глава 7* ("Другие USB-классы") содержит информацию о некоторых других USB-классах.

Главы 8—12 составляют практическую часть книги. В этой части мы создадим USB-устройства классов CDC и HID и научимся работать с ними с помощью функций Windows. Кроме того, эта часть содержит всю информацию, необходимую для создания своего драйвера.

- *Глава 8* ("Создание USB-устройства на основе AT89C5131") содержит описание процесса разработки USB-устройства. Этот проект будет базой для создания других USB-устройств.
- *Глава 9* ("Реализация класса CDC") содержит описание процесса разработки CDC-устройства и программы для обмена данными с ним.
- *Глава 10* ("Реализация класса HID") содержит описание процесса разработки HID-устройства и программы для обмена данными с ним.
- *Глава 11* ("Специальные функции Windows") содержит примеры программ, использующих функции Setup API, WMI, DirectX и множество других полезных модулей. Кроме того, в этой главе приводятся листинги программ получения Raw Input-данных.
- *Глава 12* ("Разработка USB-драйвера") описывает процесс разработки собственного USB-драйвера.

Главы 13—17 содержат справочные материалы.

Обозначения

При описании некоторых данных мы будем пользоваться битовым представлением, заключая число разрядов каждого поля в квадратные скобки. Например:

□ [5] поле А;

□ [2] поле Б.

Такое описание означает, что поле А содержит 5 битов, а поле Б — 2 бита. Еще одно представление битовых полей — указание конкретных диапазонов битов с помощью знака ":", например:

□ [16:5] зарезервированы;

□ [4:0] индекс.

Такое описание означает, что биты с 16 по 5 включительно зарезервированы, а биты с 4 по 0 включительно представляют собой индекс. Отличить первое описание от второго обычно легко по контексту изложения.

При написании чисел мы будем придерживаться следующих правил:

□ шестнадцатеричные числа будут иметь префикс "\$", например, \$45;

□ шестнадцатеричные числа могут иметь префикс "0x" или постфикс "H", если того требует контекст изложения или формат строки, например, INT 3FH;

□ битовые последовательности заключены в угловые скобки, например, <0010>, либо, при написании двоичного числа, обозначены символом "b" в конце, например, 10101111b.

Для описания версий протоколов, структур и т. д. будет использоваться специальный тип чисел BCD (Binary Coded Decimal, двоично-десятичное число). Такие числа записываются в шестнадцатеричном виде 0xJJMN для обозначения версии J.J.M.N, т. е. JJ обозначает старший номер версии, M — младший номер версии и N — номер подверсии. Например, версия 2.1.3 будет представлена числом 0x213, а версия 2.0 будет записана числом 0x0200 (см. приложение I).

При описании регистров мы будем пользоваться следующими обозначениями режимов доступа:

□ RO — (read only) регистр только для чтения, запись в него не возможна;

□ WO — (write only) регистр только для записи, чтение значения не возможно;

□ R/W — (read/write) возможно и чтение и запись значения;

□ R/W2 — (read/write word) возможно чтение и запись слова;

- R/WC — (read/write clear) разрешены как чтение, так и запись значения, однако при записи 1 в некоторый разряд регистра приводит к его сбросу в ноль.

При необходимости указания версии Windows мы будем использовать следующие сокращения:

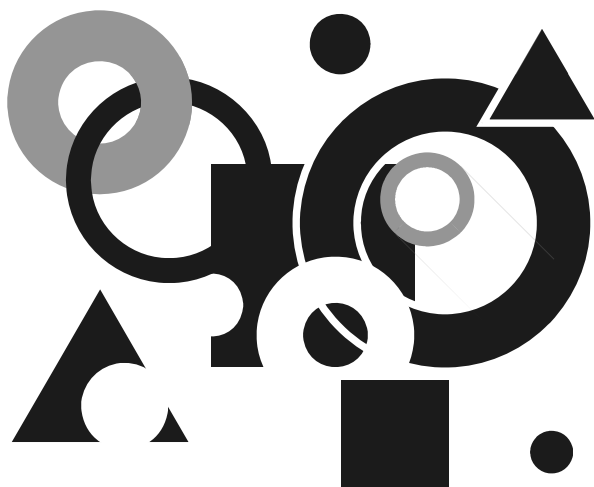
- Windows 9x будет означать семейство Windows 95/98/ME;
- Windows NT в общем случае будет обозначать семейство Windows NT/2000/XP;
- При необходимости указания конкретной версии мы будем писать номер этой версии без сокращений, например, Windows NT4 или Windows 98.

Благодарности

Вторая книга, посвященная программированию USB, не увидела бы свет, если бы не труд многих людей, прямо или косвенно принимавших участие в ее разработке. Автор выражает благодарность Сергею Малову, как обычно, помогавшему в разработке аппаратной части. Отдельное спасибо заместителю главного редактора издательства "БХВ-Петербург" Евгению Рыбакову, без участия и советов которого книга не была бы такой интересной. Спасибо всем читателям, приславшим свои отзывы, критику и пожелания, а также коллективу издательства.

Обратная связь

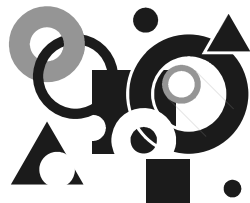
Обо всех ошибках или пожеланиях сообщайте по адресу books@pvasoft.com. На сайте <http://book.pvasoft.com> можно найти текущий список опечаток и дополнений.



ЧАСТЬ I

ОБЩИЕ СВЕДЕНИЯ О USB

Глава 1



Спецификация USB

Подробное описание USB [4] мы рассматривать не будем, а приведем лишь минимум сведений, необходимых для практической работы.

1.1. Что такое USB и зачем это надо

Шина USB (Universal Serial Bus, универсальная последовательная шина) появилась в начале 1996 года как попытка решения проблемы множественности интерфейсов. К тому времени персональные компьютеры (ПК) были оснащены большим количеством разнообразных внешних интерфейсов, полезных и необходимых, но обладающих одним недостатком: все они требовали своего специального разъема и, чаще всего, выделенного аппаратного прерывания (IRQ, Interrupt ReQuest).

Первая спецификация (версия 1.0) USB была опубликована в начале 1996 года, а осенью 1998 года появилась спецификация 1.1, исправляющая проблемы, обнаруженные в первой редакции. Весной 2000 года была опубликована версия 2.0, в которой предусматривалось 40-кратное повышение пропускной способности шины. Так, спецификация 1.0 и 1.1 обеспечивает работу на скоростях 12 Мбит/с и 1,5 Мбит/с, а спецификация 2.0 — на скорости 480 Мбит/с. При этом предусматривается обратная совместимость USB 2.0 с USB 1.x, т. е. "старые" USB 1.x устройства будут работать с USB 2.0 контроллерами, правда, на скорости 12 Мбит/с.

Разработчики шины ориентировались на создание интерфейса, обладающего следующими свойствами:

- легко реализуемое расширение периферии ПК;
- дешевое решение, позволяющее передавать данные со скоростью до 12 Мбит/с (480 Мбит/с для USB 2.0);
- полная поддержка в реальном времени голосовых, аудио- и видеопотоков;

- гибкость протокола смешанной передачи изохронных данных и асинхронных сообщений;
- интеграция с выпускаемыми устройствами;
- охват всевозможных конфигураций и конструкций ПК;
- обеспечение стандартного интерфейса, способного быстро завоевать рынок;
- создание новых классов устройств, расширяющих ПК.

Спецификация USB определяет следующие функциональные возможности интерфейса:

- простота использования для конечного пользователя:
 - простота кабельной системы и подключений;
 - скрывание подробностей электрического подключения от конечного пользователя;
 - самоидентифицирующиеся устройства с автоматическим конфигурированием;
 - динамическое подключение и переконфигурирование периферийных устройств;
- широкие возможности работы:
 - пропускная способность от нескольких Кбит/с до нескольких Мбит/с;
 - поддержка одновременно как изохронной, так и асинхронной передачи данных;
 - поддержка одновременных операций со многими устройствами (multiple connections);
 - поддержка до 127 устройств на шине;
 - передача разнообразных потоков данных и сообщений;
 - поддержка составных устройств (периферийных устройств, выполняющих несколько функций);
 - низкие накладные расходы передачи данных;
- равномерная пропускная способность:
 - гарантированная пропускная способность и низкие задержки голосовых и аудиоданных;
 - возможность использования всей полосы пропускания;
- гибкость:
 - поддержка разных размеров пакетов, которые позволяют настраивать функции буферизации устройств;
 - настраиваемое соотношение размера пакета и задержки данных;
 - управление потоком (flow control) данных на уровне протокола;

- ❑ надежность:
 - контроль ошибок и восстановление на уровне протокола;
 - динамическое добавление и удаление устройств прозрачно для конечного пользователя;
 - поддержка идентификации неисправных устройств;
 - исключение неправильного соединения устройств;
- ❑ выгода для разработчиков:
 - простота реализации и внедрения;
 - объединение с архитектурой Plug and Play;
- ❑ дешевая реализация:
 - дешевые каналы со скоростью работы до 1,5 Мбит/с;
 - оптимизация для интеграции с периферией;
 - применимость для реализации дешевой периферии;
 - дешевые кабели и разъемы;
 - использование выгодных товарных технологий;
- ❑ возможность простого обновления.

Практически все поставленные задачи были решены, и весной 1997 года стали появляться компьютеры, оборудованные разъемами для подключения USB-устройств.

1.1.1. Общая архитектура USB

Обычная архитектура USB подразумевает подключение одного или нескольких *USB-устройств* к компьютеру (рис. 1.1), который в такой конфигурации является главным управляющим устройством и называется *хостом*. Подключение USB-устройств к хосту производится с помощью *кабелей*. Для соединения компьютера и USB-устройства используется *хаб*. Компьютер имеет встроенный хаб, называемый *корневым хабом*.

1.1.2. Физическая и логическая архитектура USB

Физическая архитектура USB определяется следующими правилами (рис. 1.2):

- ❑ устройства подключаются к хосту;
- ❑ физическое соединение устройств между собой осуществляется по топологии многоярусной звезды, вершиной которой является *корневой хаб*;
- ❑ центром каждой звезды является хаб;

- каждый кабельный сегмент соединяет между собой две точки: хост с хабом или функцией (*см. далее*), хаб с функцией или другим хабом;
- к каждому порту хаба может подключаться периферийное USB-устройство или другой хаб, при этом допускается до 5 уровней каскадирования хабов, не считая корневого.

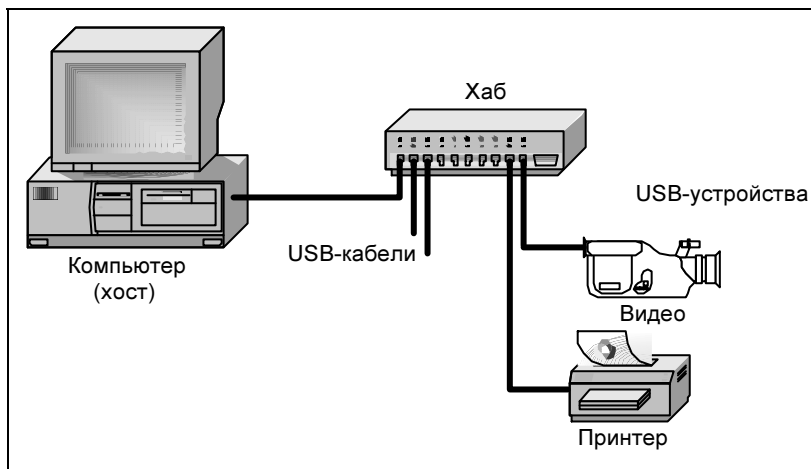


Рис. 1.1. Обычная архитектура USB

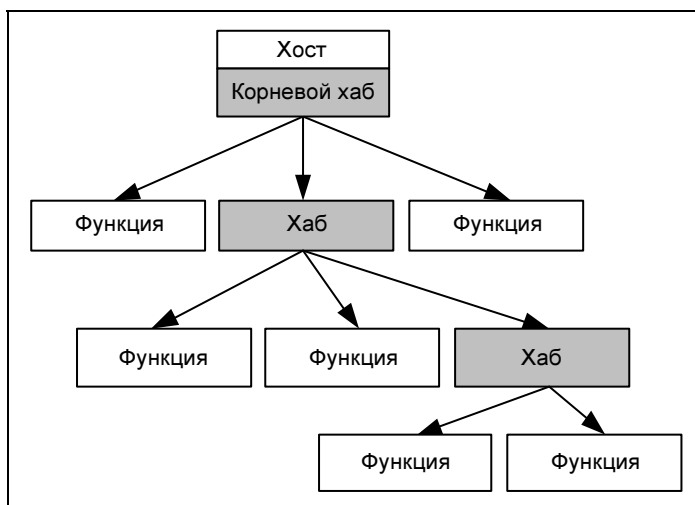


Рис. 1.2. Физическая архитектура USB

Детали физической архитектуры скрыты от прикладных программ в системном программном обеспечении (ПО), поэтому *логическая архитектура* выглядит как обычная звезда, центром которой является прикладное ПО, а вершинами — набор *конечных точек* (рис. 1.3).

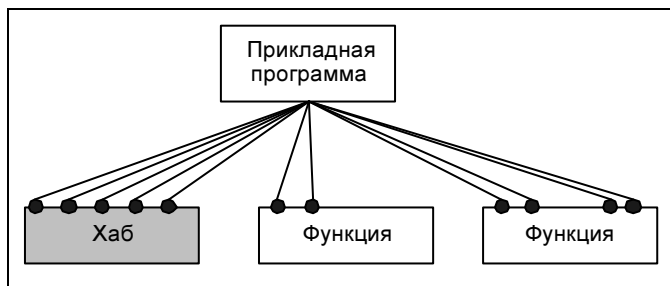


Рис. 1.3. Логическая архитектура USB

Прикладная программа ведет обмен информацией с каждой конечной точкой.

1.1.3. Составляющие USB

Шина USB состоит из следующих элементов:

- *хост-контроллер* (host controller) — это главный контроллер, который входит в состав системного блока компьютера и управляет работой всех устройств на шине USB. Для краткости мы будем писать просто хост. На шине USB допускается наличие только одного хоста. Системный блок персонального компьютера содержит один или несколько хостов, каждый из которых управляет отдельной шиной USB;
- *устройство* (device) может представлять собой хаб, функцию или их комбинацию (compound device);
- *порт* (port) — точка подключения;
- *хаб* (hub, другое название — *концентратор*) — устройство, которое обеспечивает дополнительные порты на шине USB. Другими словами, хаб преобразует один порт (*восходящий порт*, upstream port) во множество портов (*нисходящие порты*, downstream ports). Архитектура допускает соединение нескольких хабов (не более 5). Хаб распознает подключение и отключение устройств к портам и может управлять подачей питания на порты. Каждый из портов может быть разрешен или запрещен и сконфигурирован на полную или ограниченную скорость обмена. Хаб обеспечивает изоляцию сегментов с низкой скоростью от высокоскоростных. Хаб может ограничивать ток, потребляемый каждым портом;

- *корневой хаб* (root hub) — это хаб, входящий в состав хоста;
- *функция* (function) — это периферийное USB-устройство или его отдельный блок, способный передавать и принимать информацию по шине USB. Каждая функция предоставляет конфигурационную информацию, описывающую возможности периферийного USB-устройства и требования к ресурсам. Перед использованием функция должна быть сконфигурирована хостом — ей должна быть выделена полоса в канале и выбраны опции конфигурации;
- *логическое USB-устройство* (logical device) представляет собой набор конечных точек.

1.1.4. Свойства USB-устройств

Спецификация USB достаточно жестко определяет набор свойств, которые должно поддерживать любое USB-устройство:

- адресация — устройство должно отзываться на назначенный ему уникальный адрес и только на него;
- конфигурирование — после включения или сброса устройство должно предоставлять нулевой адрес для возможности конфигурирования его портов;
- передача данных — устройство имеет набор конечных точек для обмена данными с хостом. Для конечных точек, допускающих разные типы передач, после конфигурирования доступен только один из них;
- управление энергопотреблением — любое устройство при подключении не должно потреблять от шины ток, превышающий 100 мА. При конфигурировании устройство заявляет свои потребности тока, но не более 500 мА. Если хаб не может обеспечить устройству заявленный ток, устройство не будет использоваться;
- приостановка — USB-устройство должно поддерживать приостановку (suspended mode), при которой его потребляемый ток не превышает 500 мкА. USB-устройство должно автоматически приостанавливаться при прекращении активности шины;
- удаленное пробуждение — возможность удаленного пробуждения (remote wakeup) позволяет приостановленному USB-устройству подать сигнал хосту, который тоже может находиться в приостановленном состоянии. Возможность удаленного пробуждения описывается в конфигурации USB-устройства. При конфигурировании эта функция может быть запрещена.

1.1.5. Принципы передачи данных

Механизм передачи данных является асинхронным и блочным. Блок передаваемых данных называется *USB-фреймом* или *USB-кадром* (см. разд. 1.1.11) и передается за фиксированный временной интервал. Оперирование командами и блоками данных реализуется при помощи логической абстракции, называемой *каналом* (см. разд. 1.1.13). Внешнее устройство также делится на логические абстракции, называемые *конечными точками* (см. разд. 1.1.12). Таким образом, канал является логической связкой между хостом и конечной точкой внешнего устройства. Канал можно сравнить с открытым файлом.

Для передачи команд (и данных, входящих в состав команд) используется канал по умолчанию, а для передачи данных открываются либо *потокосовые каналы*, либо *каналы сообщений* (см. разд. 1.1.13).

Все операции по передаче данных по шине USB инициируются хостом. Периферийные USB-устройства сами начать обмен данными не могут. Они могут только реагировать на команды хоста.

1.1.6. Механизм прерываний

Для шины USB настоящего механизма прерываний (как, например, для последовательного порта) не существует. Вместо этого хост опрашивает подключенные устройства на предмет наличия данных о прерывании. Опрос происходит в фиксированные интервалы времени, обычно каждые 1–32 мс. Устройству разрешается посылать до 64 байт данных.

С точки зрения драйвера, возможности работы с прерываниями фактически определяются хостом, который и обеспечивает поддержку физической реализации USB-интерфейса.

1.1.7. Режимы передачи данных

Пропускная способность шины USB, соответствующей спецификации 1.1, составляет 12 Мбит/с (т. е. 1,5 Мбайт/с). Спецификация 2.0 определяет шину с пропускной способностью 400 Мбайт/с. Полоса пропускания делится между всеми устройствами, подключенными к шине.

Шина USB имеет три режима передачи данных:

- низкоскоростной (LS, Low-speed);
- полноскоростной (FS, Full-speed);
- высокоскоростной (HS, High-speed, только для USB 2.0).