

Введение в разработку HID устройств.

*Наличие встроенного в Windows 98 и Win2000 USB HID драйвера
делает разработку нового Embedded USB устройства намного
проще чем когда либо.*

Это техническое описание включает в себя все, что вам необходимо знать при разработке собственного HID-совместимого устройства. В статье изложены материалы объясняющие разработку интерфейса между приложением выполняющимся на компьютере и Embedded HID устройством.

Вся информация, представленная в этом документе, может быть использована только для ознакомительных целей. Ни одна из глав этого документа не может быть использована в какой либо форме без предварительного согласия с **KS Labs**.

Введение

Если Вы разрабатываете устройство, которое подключается к персональному компьютеру, то наверное вы уже обратили внимание на USB шину. Она как раз разработана для замены стандартных PC портов: параллельного LPT и последовательного PS/2 и RS-232. Стандартные периферийные устройства, такие как: мышь, клавиатура, жесткие диски, принтер, и сканер имеют уже стандартные USB драйвера во многих операционных системах. USB шина становится также очень популярной для многоканальных устройств сбора данных, измерительных приборов и устройств мониторинга и управления.

Для многих специализированных устройств USB HID класс предоставляет быстрый путь для разработки устройства. Хотя по спецификации, HID устройство – это устройство связи с пользователем или для простоты понимания, устройство ввода (мышь, клавиатура, джойстик) однако, HID позволяет написать низкоскоростной двусторонний обмен с другим устройством, не подпадающим под жесткое определение устройства ввода.

Поскольку Windows 98 и 2000, а также XP имеют встроенные HID-Class драйвера, у вас отпадает необходимость в трудоемкой собственной разработке драйвера для вашего нового устройства. С точки зрения Embedded программиста, чтобы определить устройство как HID, вам необходимо поддержать ряд структур описывающих HID интерфейс, а также написать алгоритм обмена по interrupt каналу передачи данных.

Основным ограничением HID устройств является скорость. Максимально достигаемая скорость составляет 64кВ/с. А это значительно меньше чем скорость всей USB шины –12МБит/с, хотя для многих приложений, например управления и индикации, вполне достаточно.

Эта статья является предисловием к разработке HID устройства. В ней будут описаны основные требования для того, чтобы операционная система обнаружила ваше устройство как HID-совместимое. Также будут представлены API функции доступные в среде Windows для связи с HID устройством и несколько бесплатных инструментальных средств, которые помогут Вам в разработке программы.

“HID Class” - один из первых USB классов, поддерживаемых операционной системой Windows. И это именно потому, что основу класса составляют устройства ввода: клавиатуры, мыши, джойстики, ТачПэды, и многое другое. Спецификация для HID класса, а также другие дополнительные документы и инструментальные средства доступны на официальном сайте: www.usb.org .

Термин “Интерфейс связи с пользователем” взят именно из-за того, что устройства HID класса непосредственно вводят в компьютер данные предоставляемые человеком. Примерами HID устройств могут служить не только клавиатуры и мыши, а и панели индикации, пульта дистанционного управления, вспомогательные телефонные клавиатуры, и др.

Но HID устройство не обязательно должно иметь кнопки, переключатели или вал-кодеры. В спецификации упоминаются и считыватели штрих-кода, и термометры, и вольтметры и другие устройства, которые могут подпадать под определение HID класса.

HID устройство кроме ввода данных в компьютер может и получать их от него. Примерами таких устройств могут служить дистанционные дисплеи, роботы и устройства управляющиеся виртуальной панелью на компьютере.

Короче говоря, HID устройством может быть любым устройством, которое может функционировать по закону определенному спецификацией. Основные особенности и ограничения HID устройств:

- Полноскоростное HID устройство может передавать вплоть до 64,000 байт в секунду (64 байта в каждом кадре 1мс). Для низкоскоростного устройства установлена скорость передачи данных только 800 байт в секунду (8 байт каждые 10мс).
- HID устройство может установить частоту своего опроса для выяснения, имеет ли устройство новые данные для пересылки.
- Весь обмен с HID устройством происходит с помощью определенной структуры, которая называется репортом. Один репорт может содержать до 65,535 байт данных. Программа микроконтроллера должна содержать дескриптор репорта "Report Descriptor", который описывает структуру данных репорта. Репорт имеет достаточно гибкую структуру для описания любого типа устройства и формата передачи данных.

Требования предъявляемые к HID устройству

Во многих отношениях HID не имеет никаких особенных отличий от других USB устройств. Каждое периферийное USB устройство должно содержать интеллектуальный контроллер, который знает как отвечать на запросы и другие события USB шины.

Контроллер должен иметь одну или более конечных точек, являющиеся буферами, которые либо принимают поток данных в направлении от ХОСТА, либо содержат данные для отправки по направлению в ХОСТ. Каждая конечная точка имеет свой номер и направление передачи. Конечная Точка 0 двунаправленная и необходима для 'Control Transfers' используемых в процессе энумерации устройства. Каждое USB устройство должно отвечать на ряд управляющих запросов, определенных спецификацией. Например, когда ХОСТ посылает запрос Get_Status к конечной точке, устройство должно вернуть информацию о состоянии конечной точки в определенном формате. Каждое устройство должно содержать соответствующие дескрипторы (Дескриптор устройства, конфигурации, интерфейса, и т.д.), которые ХОСТ должен запросить в процессе энумерации.

Кроме требований, которые относятся ко всем USB устройствам, HID имеет ряд дополнительных требований:

- HID устройство должно иметь ***Interrupt In*** конечную точку для выдачи данных в ХОСТ. ***Interrupt Out*** конечная точка для получения периодических данных от ХОСТА, является опциональной и может не использоваться.
- HID устройство должно содержать дескриптор класса "*Device Class Descriptor*" и один или более дескрипторов репорта "*HID Report Descriptor*"
- HID устройство должно поддерживать специфический для класса управляющий запрос *Get_Report*, а также опционально поддерживать дополнительный запрос *Set_Report*.
- Для ***Interrupt IN*** передачи (данные из устройства в ХОСТ), устройство должно положить данные репорта в FIFO соответствующей конечной точки и разрешить передачу. Для ***Interrupt OUT*** передачи (данные из ХОСТА в устройство), устройство должно разрешить соответствующую Out конечную точку, а затем, после прихода пакета, забрать данные из FIFO.

Выдача дескрипторов

Компьютер идентифицирует подсоединяемое периферийное USB устройство когда ХОСТ извлекает серию дескрипторов в ходе процесса эnumерации. Процесс эnumерации начинается при подключении устройства к шине, или по включению питания при уже подключенном USB устройстве.

Каждое USB устройство имеет один дескриптор устройства и один или более дескрипторов конфигурации. Если у устройства имеется несколько дескрипторов конфигурации, то ХОСТ выбирает один из них в процессе эnumерации. Каждый дескриптор конфигурации, в свою очередь, поддерживает один или несколько интерфейсных дескрипторов. В дескрипторе интерфейса область кода класса устройства должна быть установлена в 3. За дескриптором интерфейса следует дескриптор HID класса, который и определяет количество следующих за ним дескрипторов конечных точек.

Для каждого потока данных (IN или OUT) использующего "*Interrupt Transfers*" необходим свой дескриптор конечной точки. Передачи управления "*Control Transfers*" используют в качестве канала "*Default Control Pipe*", связанный с нулевой конечной точкой. При этом дескриптор конечной точки не требуется. Дескриптор для конечной точки прерывания "*Interrupt Endpoint*" определяет её номер, направление передачи данных, тип передачи (Interrupt), размер максимального пакета для каждой транзакции (от 1 до 64 байт для полноразмерных устройств и от 1 до 8 байт для низкоскоростных), а также максимальное время ожидания между транзакциями.

HID устройство должно иметь один или более дескрипторов репорта которые запрашиваются только после того, как ХОСТ определил что подключенное USB устройство относится к HID классу.

Менеджер Устройств ОС Windows использует .inf файлы чтобы решить какой драйвер назначить найденному устройству. HID устройство может использовать встроенный в ОС .inf файл для HID (hiddev.inf для Windows 98 и input.inf для Windows 2000), альтернативно, Вы можете использовать свой собственный .inf файл в котором находятся Ваши VID и PID. Преимущество собственного .inf файла состоит в том, что Вы можете включить в файл описание устройства и имя изготовителя, которые появятся на Панели Управления вместо общего термина "*Стандартное Устройство*".

HID устройство может посылать и получать репорты через управляющую конечную точку ноль или через *Interrupt* конечные точки. HID не поддерживает *Isochronous* и *Bulk* типы передачи данных.

Управляющий тип передачи данных "*Control Transfers*" не имеет гарантированного времени доставки данных. Это значит, что ХОСТ-контроллер выдает пакет данных только тогда, когда на шине есть для этого время, поэтому выдача данных может быть задержана до тех пор пока на шине не освободится необходимая полоса для их передачи. Десять процентов ширины полосы шины резервируется для управляющих передач, но ХОСТ может распределить это время для обслуживания других устройств, активных на шине в данный момент.

"*Interrupt Transfers*" имеют гарантированное максимальное время ожидания или время между двумя смежными транзакциями (установленное в дескрипторе конечной точки). Каждая транзакция несет пакет данных. Конечная Точка может запросить время опроса от 1мс до 255мс для полноскоростных устройств, или от 10мс до 255мс для низкоскоростных устройств. Например, если время опроса конечной точки составляет 10мс, то ХОСТ может провести новую транзакцию в любой момент, в интервале времени от 1мс до 10мс, после последней транзакции.

Тип (*Control* или *Interrupt*) передачи данных используемый ХОСТом для выполнения транзакции зависит от типа репорта, а также от версии Windows и от самого устройства. По HID спецификации, обмен между ХОСТом и устройством может производиться с использованием трех видов репорта: *Input*, *Output* и *Feature*. *Input* и *Output* репорты используются для передачи/приема периодических данных, например нажатие клавиш.. *Feature* репорты используются там где очень важно время доставки: для установки различных параметров устройства и его инициализации.

Когда ХОСТ запрашивает *Input* репорт, устройство выдает пакет данных с помощью *Interrupt* типа передачи данных. Периодичность генерации *Input* запросов указывается устройством в дескрипторе конечной точки.

При генерации *Output* репорта ХОСТ посылает данные в устройство используя *Control* или *Interrupt* тип передачи данных. Возможность создавать *Interrupt Output* репорт была добавлена в версию 1.1 HID спецификации, и доступна только в Windows 98 SE и последующих версиях Windows, включая Win2000 и WinXP.

Если HID интерфейс имеет *Interrupt Out* конечную точку, то Windows 98 SE или более поздняя версия Windows будет использовать ее для выдачи *OUT* репортов, а Windows 98 Gold будет использовать для выдачи *OUT* репортов только *Control Transfers*. Если HID интерфейс не имеет *Interrupt Out* конечной точки, то драйвер ОС будет использовать *Control Transfers* для выдачи *OUT* репортов.

Feature репорты имеют направление передачи данных как от ХОСТа к устройству, так и от устройства в ХОСТ. Для них всегда используется только управляющий тип передачи данных (*Control Transfers*). Для того, чтобы послать *Feature* репорт, ХОСТ инициирует запрос *Set_Report*, предшествующий пакету данных, а далее в фазе статуса ХОСТ принимает подтверждение от устройства об успешном либо не успешном принятии данных. Для того чтобы получить *Feature* репорт, ХОСТ инициирует запрос *Get_Report*, устройство при этом отвечает пакетом данных, а в фазе статуса ХОСТ возвращает в устройство информацию об успешно проведенной транзакции.

Другое преимущество *Feature* репортов - это возможность задавать каждому репорту его номер, так называемый *Report ID*. При этом у программиста появляется возможность мультиплексировать запросы, если существует необходимость создания интерфейса передачи команд управления и данных через нулевую конечную точку.

Инструментальные Средства

Для того, чтобы помочь в написании HID устройства USB Форум разработал две свободно-распространяемые программы: *HID Descriptor Tool* и *USB Compliance Tool*.

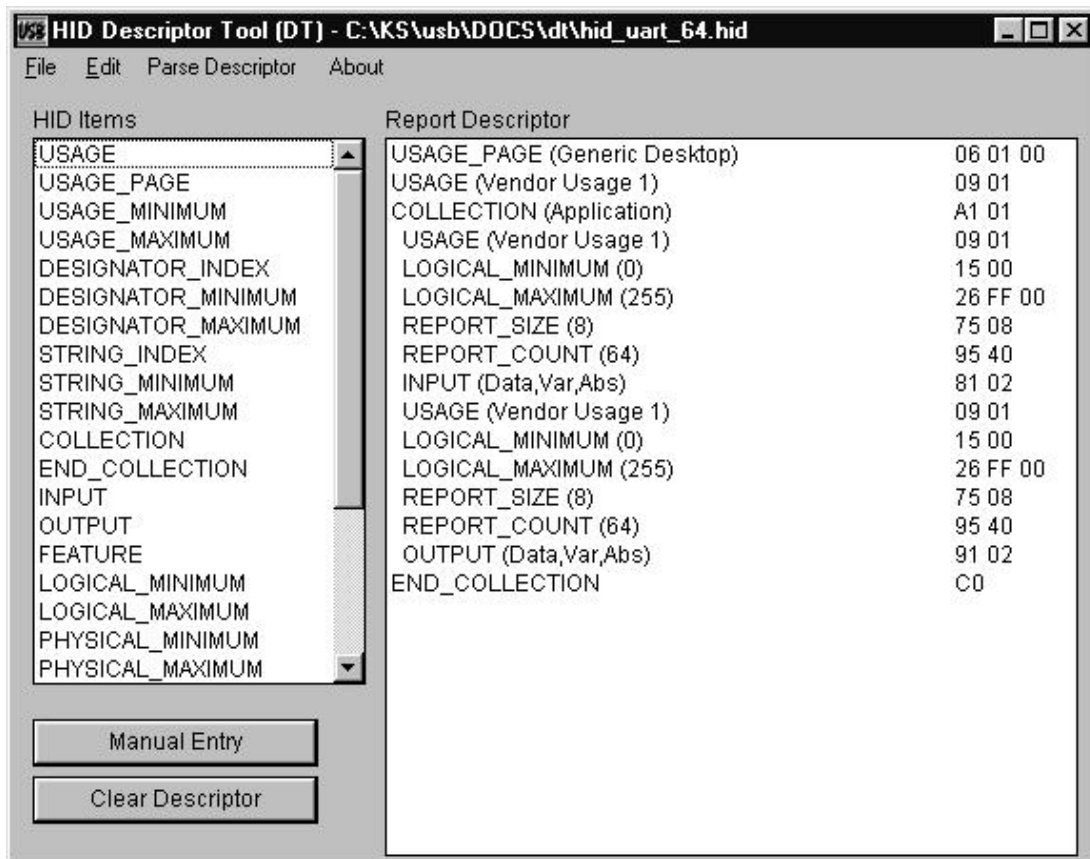


Рисунок 1: HID Descriptor Tool.

HID Descriptor Tool (Рисунок 1) позволяет автоматизировать процесс написания дескриптора репорта, а также проверить его правильность прежде чем скопировать дескриптор в устройство. Вы можете сформировать дескриптор выбирая пункты из меню, присваивая им необходимые значения (например, количество байт в поле *Report Count*). После создания дескриптора, Вы можете проверить его правильность, выбрав пункт *Parse Descriptor*. Утилита *HID Descriptor Tool* поставляется с примерами стандартных устройств для облегчения понимания HID интерфейса.

USB Compliance Tool представляет собой набор инструментальных средств позволяющих выполнить серию основных тестов на любом USB устройстве, плюс содержит дополнительные тесты для HID устройств, Концентраторов и устройств связи.

USB Compliance Tool загружает свой собственный диагностический драйвер для испытываемого устройства. Вы можете считать и проверить дескрипторы испытываемого устройства, а также выбрать и послать в устройство стандартный набор запросов.

Взаимодействие USB HID устройства с Windows

Когда Вы наконец разработали аппаратную часть HID устройства, и оно успешно было найдено операционной системой, наступило время написать программу связи между компьютером и устройством. Эта статья концентрирует внимание на ОС Windows. Поддержка HID на уровне системных драйверов доступна и в других ОС таких как Linux, MacOS. Ниже приведен набор API функций для работы с HID устройством.

HID API functions in Windows		
API function	Documentation	Use in HID communication
HidD_GetHidGuid	Windows 98/2000 DDK	Returns the GUID associated with HID
SetupDiGetClassDevs	Windows NT/2000 DDK and NT/2000 Platform SDK	Returns an array of structures containing information about all installed HID
SetupDiEnumDeviceInterfaces	Windows NT/2000 DDK and NT/2000 Platform SDK	Returns a pointer to a structure that identifies an interface in the array returned by SetupDiGetClassDevs
SetupDiGetDeviceInterfaceDetail	Windows NT/2000 DDK and NT/2000 Platform SDK	Returns a device pathname for a specified device interface
CreateFile	Windows Platform SDK	Opens a handle to a HID using the pathname returned by SetupDiGetDeviceInterfaceDetail
HidD_GetAttributes	Windows 98/2000 DDK	Returns the Vendor ID, Product ID, and Version for a specified HID
HidD_GetPreparsedData	Windows 98/2000 DDK	Returns a handle to a buffer with information about a device's capabilities
HidD_GetCaps	Windows 98/2000 DDK	Returns a structure describing a device's capabilities
HidD_GetValueCaps	Windows 98/2000 DDK	Returns a structure describing the values in a device port
HidD_GetButtonCaps	Windows 98/2000 DDK	Returns a structure describing buttons in a device report
HidD_GetFeature	Windows 98/2000 DDK	Reads a feature report from a specified HID
HidD_SetFeature	Windows 98/2000 DDK	Sends a feature report to a specified HID
ReadFile	Windows Platform SDK	Reads in input report from a specified HID
WriteFile	Windows Platform SDK	Sends out output report to a specified HID
HidP_SetButtons	Windows 98/2000 DDK	Sets the button data in a report

Таблица 1: Набор HID API функций для Windows.

DDK для Windows 98 и 2000 включает довольно полную документацию и примеры для взаимодействия с HID устройством. Примеры используют синтаксис языка C, но с некоторой доработкой и переводом, Вы можете использовать Visual Basic, Delphi, или любой другой язык который может вызывать API функции.

Функции API, которые идентифицируют устройство и возвращают путь к устройству являются функциями управления. Их описание представлено в Windows NT and 2000 SDK.

Как только получен путь к устройству, можно использовать функцию *CreateFile* чтобы открыть *Handle* на устройство, а затем использовать HID API функции для чтения *Vendor* и *Product ID*, а также информации из строкового дескриптора. Как только Вы идентифицировали устройство которое искали, можно свободно обмениваться с устройством *Input* и *Output* репортами с помощью стандартных функций: *ReadFile()* и *WriteFile()* (соответственно для входных и выходных репортов) или *HidD_Get_Feature()* и *HidD_Set_Feature()* (для *Feature* репортов).

Есть одно предостережение при использовании функции *ReadFile()*. При ее вызове пользовательская программа проваливается в системный HID драйвер и будет находиться там до тех пор, пока не получит от HID устройства запрошенное количество данных. Не помогает даже использование функции *ReadFileEx()*. Вам необходимо разместить вызов процедуры *ReadFile()* именно в том месте программы, где-бы она не “вешала” основное приложение при ожидании данных от устройства.