An Analysis of Wireless Device Implementations on Universal Serial Bus

6/3/97

Abstract

Universal Serial Bus (USB) is a new personal computer (PC) interconnect that can support simultaneous attachment of multiple devices. The USB Human Interface Device (HID) class is targeted a relatively low speed user input devices like keyboards, mice and joysticks. HID class device developers wish to provide wireless versions of their products and take advantage of the USB features. This paper presents an analysis of the issues related to implementing HID class devices that attach to the Universal Serial Bus through a USB to Wireless Adapter or Dongle. In general the issues raised here apply to any single USB device that presents multiple HID class devices to the system and the requirements of a device to be HID class compliant.

Introduction

A USB HID class device must meet the requirements of the USB Core and HID Class specifications. These specifications define the protocol, commands and data that must pass between the USB host adapter and the device. The premise of this paper is that there should be no difference between a wireless and a wired HID class device, as seen by the USB host. The function of the Wireless Dongle is to is to provide this transparent operation. This paper analyzes the requirements that a wireless device must meet to emulate a HID class device to a USB host over a wireless link.

General Wireless Requirements

A key assumption is that the Wireless Dongle communicate with up to 15 independent wireless device simultaneously and attach to the USB with a single physical connection. In essence the Wireless Dongle is a host to multiple wireless devices just as the USB Host adapter is a host to multiple USB devices. This hierarchy mirrors the current USB Host/hub/device hierarchy.

Wireless devices must also support other features of USB such as Hot Plug-in, persistent addressing, power management and wakeup. Each of these features are key to meeting the usage model that defined USB and through the Dongle, wireless devices will be required to support them.

Hot Plug-in and Device Enumeration

USB devices can be plugged into the system at any time. The USB protocol handles the enumeration of a device and the notification to the system that it has arrived. Wireless devices will be required to support the same features.

Persistent Addressing

USB provides two methods of identifying devices: serial numbers and port addresses. Serial numbers are preferred because they always follow the device however they are optional. If a serial number is not provided by the devices the device's port address is used. Using the latter method, as long as the USB topology connecting a device to the host is undisturbed, a device will receive the same address every time the system comes up. This feature allows an application to be setup once with the expectation that the same devices will exist each time the application is run. Wireless devices must provide a similar feature.

Power Management and Wakeup

USB supports a variety of power modes: On, Suspend, and Off. USB devices can be placed in suspend mode and still retain the ability to wakeup the system. The Wireless Dongle should support similar power control and conservation, and provide a mechanism for the wireless devices to wakeup the system.

Boot Device support

Wireless keyboards and mice will be required to support the same boot functionality as wired USB devices do today. Therefore the dongle and associated wireless devices implemented with the Common Class approach will have to do so as well.

USB Interface Options

There are four methods that a single USB device can use to present multiple devices to a host: Compound, Composite, Complex and Common Class.

Compound Device

A compound device looks like a hub to the system. The individual devices supported by a compound device appear to the system as devices that are attached or detached from the ports on the hub. The key difference between a compound device and a hub is that devices are made to attach or detach electronically rather than by physically inserting or removing a cable.

The advantage of this approach is that full software support is in place. The attachment and removal of wireless devices would the same to the system as the attachment and removal of wired devices on a standard hub. The disadvantage is that additional silicon would be required to present an 15 port compound device. The only silicon implementations that currently exist present a hub with one electronically attachable and detachable device and 3 or 4 external downstream ports.

Composite

Composite devices appear to the system as a single USB address that uses multiple interfaces to present the individual devices. A HID class device requires 2 endpoints: Control and Interrupt In. When implemented as a Composite Device the control endpoint (0) is shared by all the devices while separate interfaces need to be allocated to contain each interrupt endpoint.

The advantage of this approach is that full software support is in place. The disadvantage is that individual devices can only be attached or removed by forcing a disconnect, which takes down all the devices on the dongle, and reconnecting with new Configuration, Interface, and Endpoint Descriptors that defines the new set of devices attached to the dongle. The USB Endpoint Address is a 4 bit field which limits the number of HID devices that can be supported by a composite implementation to 15. Typical silicon implementations support 4 endpoints therefore only 3 HID class devices.

Complex
Complex devices are defined only by the HID class. In the HID Report Descriptor a complex device is created by declaring multiple top level Application Collections. This approach is similar to a Composite implementation however all devices defined in the Complex device share the same interrupt endpoint.

The advantage of this approach is that full software support is in place. The disadvantage is that individual devices can only be attached or removed by forcing a disconnect, which takes down all devices on the dongle, and reconnecting with a new Report Descriptor that defines the new set of devices attached to the dongle. Another disadvantage is that a Complex Device is only defined with-in the HID specification. A general purpose solution should not assume that all wireless devices will be HID devices.

Compound, Composite and Complex Implementation Summary
Assuming that the wireless dongle would support 15 devices and given the constraints of currently available hardware, some of these approaches are more expensive or easier to implement than others. For a Compound device a vendor would have to provide a hub and 15 SIE's, where an SIE contains at least 2 USB endpoints (receive and transmit serializers and data FIFOs), and address recognition logic. Composite devices would require 16 endpoints and a single address recognizer. In each case hardware optimizations could be made however the overall silicon costs will be high. A Complex devices could be implemented with only 2 endpoints and a single address recognizer. A drawback is that the Composite and Complex solutions provide no dynamic device support.

All of these approaches have the problem of how to inform the system of the PnP information related to the individual wireless devices. The information in the Dongle's Device Descriptor only describes the dongle itself. In the current USB implementation there is no method of identifying Vendor ID, Product ID, Interface Device Release Number, and Manufacturer, Product and Serial Number strings at the interface level. A new mechanism for reporting this information would have to be defined.

Common Class
Common Class defines a set of features, which can be used with a Composite Device to provide a flexible, general-purpose solution for wireless devices that minimizes silicon requirements. The features provided by the Common Class are:
- Dynamic Interfaces
- Shared Endpoints

A Common Class implementation of a dongle would present a dynamic interface for each wireless device that it might support. An additional static interface would be provided to notify the system of changes in the state of the dynamic interfaces. The common class defines how interfaces can share endpoints. This means that a dongle for wireless HID class devices could use the shared endpoint common class feature and the hardware would only require 2 physical endpoints: Control and Interrupt In, for n wireless devices. Shared endpoints allow 'packed' packets (e.g. data payloads from multiple interfaces in the same packet), so the use of this feature would improve USB efficiencies by combining data reports from multiple interfaces (devices) into a single USB packet.

The advantage of this method is that the common class was designed for this type of application. The disadvantage is that the operating system support is not yet available.

USB bandwidth consumption is also an important issue. Assuming that on the wireless side, all 15 wireless devices have an opportunity to report over a fixed period of time. Then to ensure no loss of data, during that same period of time on the USB side, 15 transactions must take place to poll the respective interrupt endpoints that represent those devices. This constraint applies to Compound, Composite, and Complex designs.

The Common Class approach provides the best of all worlds: Composite device with Common Class features (2 endpoints, dynamic interfaces, and shared endpoints).

Common Class Requirements

The Common Class specification is not complete however the following features are required.

Boot Device support
Common Class implementations must be capable of supporting boot devices. Generally this means a fixed, low overhead method that a BIOS can use to identify and use boot devices.

New Device Notification Mechanism
A requirement of any Common Class implementation is that a standard mechanism be defined for system notification of changes in dynamic interface status. A Notification mechanism is referenced in the current version of the specification, however the implementation details are not defined.

Shared Endpoints
The Common Class specification will allow sharing of an endpoint by all interfaces that require the same function, Interrupt In, Bulk Out, etc. Data from a particular interface is placed in a wrapper that identifies the interface that it is associated with. The operating system will transparently route the data to the correct pipe.

Packed Packets
A basic assumption of this paper is that up to 15 wireless devices can be attached to the system through a single USB device (dongle). Typically this means that an individual Interrupt In transaction must be initiated on the USB for each wireless device that is active on the dongle. Shared Endpoints allow multiple data payloads per packet. This feature could be utilized to minimize USB bandwidth requirements. Using this technique the USB polling rate of dongle can be dropped to a lower rate, where each access to the dongle returns the data from multiple devices in a single packet.

PnP Information for Dynamic Interfaces
It is crucial that the vendor of the individual wireless devices be able to be identified. The common class must offer a mechanism for providing the PnP information (Vendor ID, Product ID, Device Release Number, Manufacturer String, Product String, and optional Serial Numbers information) from the device attached to each dynamic interface.

Flow Control

Flow control imposed by the USB protocol will have to be accommodated by the wireless dongle.

HID class devices use two USB endpoints: Control and Interrupt In. The Control endpoint returns data given a Get_Report or Get_Descriptor command and receives data when given a Set_Report command. Any low level flow control on the Control endpoint is strictly a function of how long the device takes to process the current command and setup for the next.

However, the Interrupt In endpoint works differently. It will NAK on read attempts until some event occurs in the device that requires it to send data to the host. A wireless HID device must be able to make this distinction in the messages that it sends to the host i.e., the immediate transfers that take place over the Control endpoint and the event driven transfers that take place over the Interrupt In endpoint.

Wireless Device Requirements

The following commands that must be supported by USB Wireless devices. See the USB Core Specification and HID Class Specification for the detailed operation of these commands.

Get_Descriptor Request
Get_Report Request
Set_Report Request
Get_Protocol Request **-** for devices that declare themselves as boot devices
Set_Protocol Request **-** for devices that declare themselves as boot devices

Followint are the USB and HID Descriptors that must be supplied by USB Wireless devices:

Device Descriptor – This information is used by the Dongle to inform the USB Host of the vendor of the wireless device.

HID Descriptor

Report Descriptor

String Descriptors – Associated with the Device and Report Descriptors

Physical Descriptors

Wireless Dongle Requirements

1) Commands emulated by the Wireless dongle:
   Get_Idle Request
   Set_Idle Request

2) USB Host notification of the arrival of new wireless devices.

3) If implemeted using Common Class features, the wireless dongle must build the Dynamic Interface, and Endpoint Descriptors from the information provided in the Device Descriptor by the wireless device.

4) Flow Control **-** The wireless dongle must provide a method of managing flow control between the USB and Wireless protocols.

5) Endpoint Routing

USB uses separate endpoints to prevent blocking of data. The HID Interrupt In endpoint (1) is used by a device to inform the system of asynchronous events. Here a NAK to a USB poll means that nothing has changed on the any of the Input fields declared in the Report Descriptor. The wireless protocol must provide a mechanism for notification of asynchronous events genreated by wireless devices. And the wireless dongle must route these messages over the Interrupt In Endpoint.

Common Class Descriptor Changes

Following are examples of the changes and features to the existing USB Descriptors that are required for the proper operation of a Wireless Dongle.

Interface Level PnP Support
Below are detailed Descriptors for a single HID device. The fields of the Interface Descriptor in double outline identify the new fields that would be required in an Interface Descriptor to support PnP and vendor identification at the Interface level.

Common Class Descriptors used by a USB-Wireless dongle

The Notification Interface is permanently connected. The individual Dynamic Interfaces are enabled at run time. They are all HID class devices and share the same endpoint. Data is packed in the packet read from the shared Interrupt endpoint.

Summary

The issues related to supporting wireless devices through USB are discussed above.

The existing options for supporting a host for wireless devices under USB are Compound, Composite or Complex implementations. The Compound device solution fits very cleanly into the existing architecture supporting all the requirements of a wireless dongle, however it is costly silicon point of view.

Composite or Complex solutions do not provide adequate support for allowing wireless devices to enter or exit the system (dynamic interfaces) or mechanisms to present a device's PnP information (vendor and product identification, serial number, etc.) to the system. However these problems are workable:

1) The current options can only provide brute force solutions to the dynamic interface problem. Alternate mechanisms could be developed, however these solutions would most likely require some level of additional system level software support as well as new HID Usage definitions.

2) Providing the PnP information could also be handled by a new HID Usage definitions that provide the equivalent information. Again, there would have to be operating system level support to read and parse this information, and start up any associated application.

Ideally, any solution should be part of the USB Core Specification rather than class specific. The Common Class features combined with those of Composite devices offers the most promising solution for USB wireless implementations. The problems Common Class are:

1) These implementations will not be possible until the Common Class document is complete.

2) Supplying the PnP information at the Interface level was not on the Common Class committee's original agenda. We will have to track their efforts to ensure that they provide a mechanism to do this.

In the end a Common Class solution will provide a generic solution that will minimize endpoint and bus bandwidth demands while maximizing flexibility.