Cách sử dụng WinUSB để giao tiếp với thiết bị USB

Ngày 30 tháng 3 năm 2009

trừu tượng

Các nhà cung cấp phần cứng độc lập (IHV) sản xuất thiết bị USB thường phải cung cấp một cách để các ứng dụng truy cập vào các tính năng của device. Về mặt lịch sử, điều này có nghĩa là sử dụng Mô hình driver Windows® (WDM) để triển khai driver function cho thiết bị và cài đặt driver trong ngăn xếp của thiết bị trên hệ thống driver được cung cấp trên hệ thống. Windows driver Foundation (WDF) hiện là mô hình ưa thích cho driver USB. Nó cung cấp cho IHV ba tùy chọn để cung cấp quyền truy cập vào thiết bị USB:

• Triển khai driver user-mode bằng cách sử dụng driver WDF user-mode framework framework (UMDF).

• Triển khai kernel-mode driver bằng cách sử dụng WDF kernel-mode driver framework (KMDF).

• Cài đặt WinUsb.sys làm driver function driver và cung cấp một ứng dụng truy cập device bằng cách sử dụng API WinUSB.

Sách trắng này cung cấp các hướng dẫn về thời điểm sử dụng từng tùy chọn và include hướng dẫn chi tiết về cách cài đặt WinUsb.sys như một driver function trên thiết bị và sử dụng API WinUSB để giao tiếp với thiết bị.

Thông tin này áp dụng cho các hệ điều hành sau: Windows 7

Windows Server® 2008 Windows Vista® Windows XP

Tài liệu tham khảo và resource được thảo luận ở đây được liệt kê ở phần cuối của bài viết này. Để biết thông tin mới nhất, xem:

<http://www.microsoft.com/whdc/connect/usb/winusb_howto.mspx>

Thông tin trong tài liệu này thể hiện quan điểm hiện tại của Tập đoàn Microsoft về các vấn đề được thảo luận kể từ ngày xuất bản. Vì Microsoft phải đáp ứng với các điều kiện thị trường thay đổi, nên không được hiểu là một cam kết về phía Microsoft và Microsoft không thể đảm bảo tính chính xác của bất kỳ thông tin nào được trình bày sau ngày xuất bản.

Trang trắng này chỉ dành cho mục đích thông tin. MICROSOFT KHÔNG ĐẢM BẢO, RPR RÀNG, NGAY LẬP TỨC HOẶC THỐNG KÊ, NHƯ CÁC THÔNG TIN TRONG TÀI LIỆU NÀY.

Tuân thủ tất cả các luật bản quyền hiện hành là trách nhiệm của người dùng. Không giới hạn các quyền theo bản quyền, không một phần nào của tài liệu này có thể được sao chép, lưu trữ hoặc đưa vào hệ thống truy xuất, hoặc truyền dưới bất kỳ hình thức nào hoặc bằng bất kỳ phương tiện nào (điện tử, cơ khí, sao chụp, ghi âm, hoặc cho bất kỳ hình thức nào) mục đích, mà không có sự cho phép rõ ràng bằng văn bản của Microsoft Corporation.

Microsoft có thể có bằng sáng chế, ứng dụng bằng sáng chế, nhãn hiệu, bản quyền hoặc quyền sở hữu trí tuệ khác liên quan đến vấn đề trong tài liệu này. Trừ khi được quy định rõ ràng trong bất kỳ thỏa thuận cấp phép bằng văn bản nào từ Microsoft, việc cung cấp tài liệu này không cung cấp cho bạn bất kỳ giấy phép nào cho các bằng sáng chế, nhãn hiệu, bản quyền hoặc tài sản trí tuệ khác.

Trừ khi có ghi chú khác, các công ty, tổ chức, sản phẩm mẫu, tên domain, địa chỉ email, logos, người, địa điểm và sự kiện được mô tả ở đây là hư cấu và không liên kết với bất kỳ công ty, tổ chức, sản phẩm, tên domain, địa chỉ email nào, logo, người, địa điểm hoặc sự kiện được dự định hoặc nên được suy luận.

© 2009 Tập đoàn Microsoft. Đã đăng ký Bản quyền.

Microsoft, MSDN, Windows, Windows Server và Windows Vista là các nhãn hiệu hoặc nhãn hiệu đã đăng ký của Tập đoàn Microsoft tại Hoa Kỳ và / hoặc các quốc gia khác.

Tên của các công ty và sản phẩm thực tế được đề cập ở đây có thể là thương hiệu của chủ sở hữu tương ứng của họ.

Lịch sử sửa đổi

Ngày thay đổi

Ngày 30 tháng 3 năm 2009 Đã thêm thông tin bổ sung để liên lạc với các endpoint.

Ngày 6 tháng 12 năm 2007 Đã thêm một phần trên DFU vào Câu hỏi thường gặp về WinUSB.

Đã xóa ghi chú khỏi Bảng 1 cho thấy WinUSB không hỗ trợ tạm dừng chọn lọc WinUSB trên Windows XP.

Ngày 30 tháng 8 năm 2007 Tạo

Nội dung

Giới thiệu 4

Tóm tắt về Khả năng của WinUSB, UMDF và KMDF 5

Nguyên tắc cung cấp quyền truy cập vào thiết bị USBs 6

Giới thiệu về WinUSB 6

Câu hỏi thường gặp về WinUSB 7

Cách cài đặt WinUsb.sys dưới dạng driver function 8

Cách sử dụng API WinUSB 12

Nhận một hand cho thiết bị và Khởi tạo WinUSB 12

Có được đường dẫn device 13

Có được file handle cho thiết bị 15

Khởi tạo WinUSB 15

Cấu hình thiết bị. 15

Giao tiếp với các endpoint. 17

Yêu cầu control 17

Yêu cầu I / O của WinUSB 19

Yêu cầu viết WinUSB 19

WinUSB Viết hành vi mặc định 20

WinUSB Đọc Yêu cầu 20

WinUSB mặc định Đọc hành vi 20

Chính sách ống 21

Quản lý năng lượng WinUSB 23

Đình chỉ chọn lọc 23

Phát hiện nhàn rỗi 24

Cân nhắc tính năng trong tương lai 24

Tóm tắt 24

Tài nguyên 24

1. Giới thiệu

Các nhà cung cấp phần cứng độc lập (IHV) sản xuất USB device thường phải cung cấp một cách để các ứng dụng truy cập vào các tính năng của device. Trong lịch sử, điều này có nghĩa là sử dụng Mô hình driver Windows® ( WDM) để triển khai driver function cho thiết bị và cài đặt driver trong ngăn xếp thiết bị trên các driver giao thức do hệ thống cung cấp như USBhub.sys hoặc USBccgp.sys. Driver function hiển thị interface device mà các ứng dụng sử dụng để lấy file device handle . Sau đó, họ có thể sử dụng handle để giao tiếp với driver bằng cách gọi các hàm function của Windows như Readfile và deviceIoControl.

driver là cách linh hoạt nhất để cung cấp quyền truy cập vào thiết bị USB và cho phép thiết bị này có thể được truy cập bởi bất kỳ ứng dụng nào, include nhiều ứng dụng đồng thời. Tuy nhiên, driver yêu cầu một nỗ lực phát triển đáng kể và một số thiết bị đủ đơn giản để chúng không yêu cầu sự hỗ trợ đầy đủ của driver function tùy chỉnh. Ví dụ: các thiết bị như các bộ điều khiển máy hoặc loggers dữ liệu thường chỉ được truy cập bởi một ứng dụng duy nhất được viết riêng cho thiết bị liên quan. Trong những trường hợp này, WinUSB cung cấp một giải pháp thay thế đơn giản hơn để triển khai driver USB tùy chỉnh.

WinUSB được phát triển đồng thời với Windows driver Foundation (WDF) và có sẵn cho Windows XP và các phiên bản Windows mới hơn. Nó include một driver kernel-mode, WinUsb.sys, là một phần không thể thiếu của WDF-mode driver framework (UMDF) hỗ trợ cho driver USB. Tuy nhiên, đối với các thiết bị USB chỉ được truy cập bởi một ứng dụng duy nhất, các nhà cung cấp thường có thể cài đặt WinUsb.sys dưới dạng thiết bị của họ fvion driver thay vì triển khai driver tùy chỉnh. Sau đó, ứng dụng có thể định cấu hình device và truy cập các endpoint của nó bằng cách sử dụng API WinUSB.

Đối với những thiết bị USB này yêu cầu các tính năng của driver function tùy chỉnh, cách tiếp cận ưa thích là WDF. Mô hình lập trình WDF và device driver interface (DDI) giúp driver USB WDF dễ dàng hơn nhiều so với driver WDM tương đương. Bạn có thể triển khai WDF USB driver theo một trong các cách sau:

• Sử dụng driver user-mode framework (UMDF) để triển khai driver USB user-mode cho hầu hết các thiết bị USB cho Windows XP trở lên.

• Sử dụng kernel-mode driver framework (KMDF) để triển khai kernel-mode USB drivers cho mọi thiết bị USB cho Windows 2000 trở lên.

Sách trắng này mô tả cách chọn cách tốt nhất để cung cấp cho các ứng dụng quyền truy cập vào thiết bị USB và trả lời một số câu hỏi phổ biến về WinUSB. Phần lớn của bài báo là một bản hướng dẫn chi tiết, include các mẫu code, cách cài đặt WinUsb.sys như một thiết bị USB, function driver và cách sử dụng API WinUSB để giao tiếp với thiết bị.

Các ví dụ trong bài viết này dựa trên Bộ công cụ học tập OSR USB FX2, nhưng bạn có thể dễ dàng mở rộng các quy trình sang các thiết bị USB khác. Hình 1 là sơ đồ đơn giản hóa của thiết bị FX2 và hiển thị các tính năng key của nó.

Tóm tắt về Khả năng của WinUSB, UMDF và KMDF

Bảng 1 tóm tắt các khả năng của driver USB WinUSB, UMDF và driver USB KMDF.

Bảng 2 tóm tắt các tùy chọn WDF nào được hỗ trợ bởi các phiên bản Windows khác nhau.

Nguyên tắc cung cấp quyền truy cập vào thiết bị USB

Nói chung, bắt đầu với cách tiếp cận đơn giản nhất, WinUSB và chuyển sang các giải pháp phức tạp hơn chỉ khi cần thiết.

Nếu device của bạn không hỗ trợ chuyển giao đẳng thời và chỉ được truy cập bởi một ứng dụng duy nhất, ứng dụng có thể sử dụng WinUSB để định cấu hình thiết bị và truy cập các endpoint của thiết bị. Ví dụ, WinUSB là phương pháp ưa thích để sử dụng cho trạm thời tiết điện tử chỉ được truy cập bởi một ứng dụng được đóng gói với thiết bị. WinUSB cũng hữu ích cho giao tiếp chẩn đoán với thiết bị và phần sụn nhấp nháy.

Một số loại device yêu cầu driver function tùy chỉnh. Ví dụ: bạn phải triển khai driver cho device được truy cập bởi nhiều ứng dụng đồng thời. Các hướng dẫn chung để triển khai driver USB WDF là như sau:

• UMDF là phương pháp ưa thích và phù hợp với hầu hết các thiết bị USB. Ví dụ: UMDF drivers là tùy chọn ưa thích cho các trình phát nhạc hoặc dongle nối tiếp.

• device với các tính năng không được UMDF hỗ trợ hoặc phải chạy trên các phiên bản Windows sớm hơn Windows XP yêu cầu driver KMDF. Ví dụ: bộ điều hợp mạng USB yêu cầu driver KMDF vì cạnh trên của driver phải giao tiếp với ngăn xếp interface mạng kernel-mode (NDIS). Giao tiếp như vậy chỉ có thể được thực hiện từ kernel mode. Driver KMDF cũng được yêu cầu cho các thiết bị có hỗ trợ chuyển giao đẳng thời.

Để thảo luận chi tiết về cách triển khai driver UMDF và KMDF, hãy xem cuốn sách Báo chí của Microsoft Phát triển driver với Windows driver Foundation hoặc trang web WHDC của Windows driver. Phần còn lại của tờ giấy trắng này được dành cho WinUSB.

Giới thiệu về WinUSB

WinUSB include hai thành phần chính:

• WinUsb.sys là driver kernel-mode có thể được cài đặt dưới dạng filter hoặc function driver, phía trên giao thức drivers trong ngăn xếp thiết bị USB kernel-mode.

• WinUsb.dll là một DLL chế độ người dùng để hiển thị API WinUSB. Các ứng dụng có thể sử dụng API này để liên lạc với WinUsb.sys khi nó được cài đặt dưới dạng driver function .

Đối với các thiết bị không yêu cầu driver function tùy chỉnh, WinUsb.sys có thể được cài đặt trong ngăn xếp của kernel, kernel-mode như driver function. user-mode processes sau đó có thể giao tiếp với WinUsb.sys thông qua một tập hợp các yêu cầu điều khiển I / O của device.

Bộ xử lý API WinUSB được WinUSB.dll đơn giản hóa quy trình giao tiếp này. Thay vì xây dựng các yêu cầu điều khiển I / O của device để thực hiện các thao tác USB tiêu chuẩn, như cấu hình thiết bị, gửi các yêu cầu điều khiển và chuyển dữ liệu đến hoặc từ các ứng dụng của device, gọi các hàm tương đương WinUSB API functions. Trong nội bộ, WinUsb.dll sử dụng dữ liệu mà ứng dụng chuyển đến WinUSB function để construct yêu cầu điều khiển I / O thích hợp của thiết bị và gửi

yêu cầu WinUsb.sys để processing. Khi yêu cầu hoàn tất, WinUSB function sẽ chuyển bất kỳ thông tin nào được trả về bởi WinUsb.sys, chẳng hạn như dữ liệu từ một yêu cầu đã đọc trở lại process.

Sử dụng API WinUSB để liên lạc với thiết bị đơn giản hơn nhiều so với triển khai driver nhưng có các hạn chế tương ứng sau:

• API WinUSB chỉ cho phép một ứng dụng tại một thời điểm giao tiếp với thiết bị. Nếu có nhiều hơn một ứng dụng có thể giao tiếp đồng thời với thiết bị, bạn phải triển khai driver function.

• API WinUSB không hỗ trợ truyền dữ liệu đến hoặc từ các endpoint đẳng thời. Truyền đồng bộ yêu cầu driver kernel-mode function.

• API WinUSB không hỗ trợ các thiết bị đã có hỗ trợ kernel-mode. Ví dụ về các thiết bị như vậy include modem và bộ điều hợp mạng, được hỗ trợ bởi API điện thoại (TAPI) và NDIS, tương ứng.

• Đối với thiết bị multifunction, bạn có thể sử dụng thiết bị INF để chỉ định kernel-mode driver trong hộp hoặc WinUsb.sys cho mỗi USB function riêng biệt. Tuy nhiên, bạn chỉ có thể chỉ định một trong các tùy chọn này cho một function cụ thể, không phải cả hai.

WinUsb.sys cũng là một phần key của liên kết giữa driver UMDF function và thiết bị liên quan. WinUsb.sys được cài đặt trong stack 8 kernel-mode của device dưới dạng driver filter phía trên. Một ứng dụng giao tiếp với driver function của driver 8 UMDF để phát hành các yêu cầu điều khiển đọc, ghi hoặc thiết bị. Driver tương tác với framework, chuyển yêu cầu tới WinUsb.sys, xử lý yêu cầu này và chuyển nó đến giao thức drivers và cuối cùng đến thiết bị . Bất kỳ phản hồi trả về bởi đường dẫn Reverse. WinUsb.sys cũng đóng vai trò là chủ sở hữu và cắm điện trên thiết bị (PPO) của thiết bị.

Câu hỏi thường gặp về WinUSB

Câu hỏi thường gặp này trả lời một số câu hỏi phổ biến về WinUSB.

Phiên bản nào của Windows hỗ trợ WinUSB?

WinUSB được hỗ trợ bởi:

• Tất cả các SKU Windows Vista.

• Tất cả các SKU khách của các phiên bản 32 bit của Windows XP SP2 và các gói dịch vụ sau này.

Lưu ý: WinUSB không có nguồn gốc từ Windows XP; nó phải được cài đặt với trình co‐installer WinUSB. Để biết chi tiết, hãy xem cách Cách cài đặt WinUsb.sys dưới dạng driver function sau này trong bài viết này.

Những tính năng USB nào được WinUSB hỗ trợ?

Bảng 3 cho thấy các tính năng USB cấp cao nào được WinUSB hỗ trợ trong Windows Vista và Windows XP.

Bảng 3. Hỗ trợ tính năng WinUSB

Tính năng Windows XP Windows Vista

device Yêu cầu điều khiển I / O được hỗ trợ Hỗ trợ

Chuyển khoản không đồng bộ Không được hỗ trợ Không được hỗ trợ

Chuyển số lượng lớn, control và ngắt được hỗ trợ

Đình chỉ chọn lọc được hỗ trợ Hỗ trợ

Đánh thức từ xa được hỗ trợ Hỗ trợ

Làm thế nào để tôi được phép sử dụng WinUSB?

WinUSB được include trong Bộ điều khiển Windows (WDK) dưới dạng gói co‐installer, WinUSBCoInstaller.dll. Các DLL riêng biệt cho các hệ thống x86 và x64 được đặt trong thư mục WinDDK \ BuildNumber \ Redist \ Winusb. Các DLL được ký kết. IHV có thể phân phối lại các DLL này.

WinUSB có hỗ trợ DFU profile cho thiết bị USB không?

Microsoft đã không triển khai driver cụ thể cho các bản cập nhật firmware. WinUSB không hỗ trợ các cổng hoạt động khởi động lại máy chủ và cổng chu kỳ. Tuy nhiên, một số thiết bị đã hiển thị mã control do nhà cung cấp xác định để bắt đầu thiết lập lại. Trong trường hợp đó, bạn có thể sử dụng WinUSB để bắt đầu thiết lập lại bằng cách gửi yêu cầu chuyển điều khiển có chứa mã điều khiển do nhà cung cấp xác định đến thiết bị. Để thảo luận về cách sử dụng WinUSB để gửi các yêu cầu chuyển điều khiển, hãy xem phần Yêu cầu điều khiển của người dùng sau trong bài viết này.

Làm cách nào để báo cáo lỗi WinUSB hoặc thực hiện các yêu cầu tính năng?

Để báo cáo lỗi hoặc thực hiện các yêu cầu tính năng, hãy sử dụng <http://support.microsoft.com> để liên hệ với Trình quản lý tài khoản kỹ thuật Microsoft hoặc Kỹ sư hỗ trợ sản phẩm của bạn.

2. Cách cài đặt WinUsb.sys dưới dạng driver function

Trước khi ứng dụng của bạn có thể sử dụng API WinUSB để liên lạc với thiết bị, bạn phải cài đặt WinUsb.sys với tư cách là driver function của device. Để làm như vậy, tạo một gói include:

• Trình co‐installer WinUSB, cài đặt WinUSB trên hệ thống đích, nếu cần.

WDK include ba phiên bản của trình co‐installer: một cho hệ thống x86, một cho hệ thống x64 và một cho hệ thống Itanium. Tất cả chúng đều được đặt tên WinUSBCoInstaller.dll và được đặt trong thư mục con thích hợp của thư mục WinDDK \ BuildNumber \ redist \ winusb.

• co‐installer KMDF, cài đặt phiên bản chính xác của KMDF trên hệ thống đích, nếu cần.

Trình co‐installer này là bắt buộc vì WinUsb.sys phụ thuộc vào KMDF. Phiên bản WinUSB mà bài báo này dựa trên phiên bản KMDF 1.5 và trình co‐installer có liên quan là WdfCoInstaller01005.dll. Các phiên bản x86 và x64 của WdfCoInstaller01005.dll được include trong WDK trong thư mục WinDDK \ BuildNumber \ redist \ wdf.

• Một INF cài đặt WinUsb.sys làm driver function của thiết bị.

• Một tập tin catalog đã ký cho gói.

File này được yêu cầu để cài đặt WinUSB trên các phiên bản x64 của Windows Vista. Để biết thêm thông tin về cách tạo và kiểm tra danh mục file đã ký, hãy xem kernel kernel-mode code Đăng nhập Walk phiên bản trên trang web WHDC.

Lưu ý: Khi các phiên bản WDF mới được phát hành, tên co‐installer sẽ thay đổi để phản ánh số phiên bản WDF. Ví dụ, phiên bản WDF 1.7 hiện đang được phát triển và trình co‐installer KMDF cho phiên bản đó được đặt tên là WdfCoInstaller01007.dll.

Ví dụ sau đây là một INF đơn giản cài đặt WinUsb.sys làm driver function cho thiết bị Fx2 :

INF này có thể được sử dụng cho hầu hết các thiết bị USB, với một số sửa đổi đơn giản. Nói chung, bạn nên thay đổi USB USB\_Install trong tên phần thành giá trị DDInstall thích hợp. Bạn cũng nên thực hiện các thay đổi đơn giản cho các phần của phiên bản, nhà sản xuất và kiểu máy, chẳng hạn như cung cấp tên nhà sản xuất phù hợp, tên của danh mục sản phẩm đã ký của bạn , lớp thiết bị chính xác và mã định danh nhà cung cấp (VID) và mã định danh nhà cung cấp (VID) cho thiết bị.

Các giá trị cụ thể của device nên được thay đổi được hiển thị trong ví dụ được in đậm. Các giá trị có thể cần phải thay đổi, chẳng hạn như các giá trị phụ thuộc vào số phiên bản, được in nghiêng.

Lưu ý: Để biết thêm thông tin về các lớp USB device, hãy xem Các lớp USB được hỗ trợ trong bộ WDK.

Ngoài các giá trị cụ thể của device và một số vấn đề được lưu ý trong danh sách sau đây, bạn có thể sử dụng các phần và directive này mà không cần sửa đổi để cài đặt WinUSB cho bất kỳ thiết bị USB nào. Các ghi chú sau tương ứng với các bình luận được đánh số trong INF.

1. Các lệnh Include và Needs trong phần USB\_Install được yêu cầu để cài đặt WinUSB trên các hệ thống Windows Vista. Các hệ thống Windows XP bỏ qua các directive này. Những directive này không nên được sửa đổi.

2. Include trong phần USB\_Install.services include INF do hệ thống cung cấp cho WinUSB. INF này được cài đặt bởi trình co‐installer WinUSB nếu nó chưa có trên hệ thống đích. Lệnh Addservice chỉ định WinUsb.sys là driver function của device. Những directive này không nên được sửa đổi.

3. Phần WinUSB\_serviceInstall chứa dữ liệu để cài đặt WinUsb.sys dưới dạng dịch vụ. Phần này không nên được sửa đổi.

4. Lệnh Kmdfservice trong phần USB\_Install.Wdf cài đặt WinUsb.sys dưới dạng kernel-mode service. Phần WinUSB\_Install được tham chiếu chỉ định phiên bản thư viện KMDF. Ví dụ này dựa trên phiên bản WindowsKv Vista Vista của WDK (bản dựng 6000), include phiên bản KMDF 1.5. Các phiên bản sau của WinUSB có thể yêu cầu phiên bản KMDF mới hơn.

5. USB\_Install.HW là phần key trong INF. Nó chỉ định interface nhận dạng duy nhất trên toàn cầu device (GUID) cho thiết bị của bạn. Lệnh AddReg đặt GUID interface vào giá trị Registry tiêu chuẩn. Khi WinUsb.sys được tải dưới dạng device, function driver, nó sẽ đọc giá trị Registry và sử dụng GUID được chỉ định để thể hiện interface device. Bạn nên thay thế GUID trong ví dụ này bằng một ví dụ mà bạn tạo riêng cho thiết bị của mình. Nếu các giao thức cho device thay đổi, bạn nên tạo một interface device mới GUID.

6. Phần USB\_Install.CoInstallers, include các phần AddReg và Copyfiles được tham chiếu, chứa dữ liệu và hướng dẫn cài đặt các trình co‐installer WinUSB và KMDF và liên kết chúng với thiết bị . Hầu hết các thiết bị USB có thể sử dụng các phần và directive này mà không cần sửa đổi.

7. Các phiên bản x86 và x64 của Windows có các trình co‐installer riêng. Ví dụ này lưu trữ chúng trên đĩa cài đặt trong các thư mục có tên i386 và amd64, tương ứng. Hình 2 (trên trang sau) cho thấy một ví dụ về những gì mà gói driver IHV có thể chứa.

Lưu ý: Mỗi trình co‐installer có phiên bản miễn phí và được kiểm tra. Sử dụng phiên bản miễn phí để cài đặt WinUSB trên các bản dựng Windows miễn phí, include tất cả các phiên bản bán lẻ. Sử dụng phiên bản đã kiểm tra, trong đó có phiên bản hậu tố trực tuyến có thể cài đặt WinUSB để cài đặt WinUSB trên các bản dựng đã kiểm tra của Windows.

INF cũng thường chứa các directive để cài đặt ứng dụng liên quan.

Bạn cài đặt WinUsb.sys chính xác như mọi driver khác. Cách tiếp cận đơn giản nhất là cắm vào device và sử dụng Trình hướng dẫn phần cứng mới hoặc Trình quản lý thiết bị để cài đặt driver bằng cách sử dụng INF được thảo luận trong phần này. Để biết thêm chi tiết về INF và cách cài đặt driver drivers, hãy xem Cài đặt device và driver trên WDK.

Hình 2. Ví dụ về nội dung gói driver

Cách sử dụng API WinUSB

Nếu một thiết bị USB có WinUsb.sys là driver function của nó, thì ứng dụng được liên kết sẽ giao tiếp với thiết bị này bằng cách gọi các function API WinUSB khác nhau. Để sử dụng API WinUSB trong một ứng dụng:

• Include WinUsb.h. Nó được include trong WDK, trong WINDDK \ BuildNumber \ inc \ ddk.

• Thêm WinUsb.lib vào danh sách các thư viện được liên kết với ứng dụng của bạn. WinUsb.lib được include trong WDK. Phiên bản dành cho Windows XP nằm trong WINDDK \ BuildNumber \ lib \ wxp \ i386. Có các phiên bản WinUsb.lib riêng cho Windows Vista cho mỗi kiến trúc CPU được hỗ trợ. Chúng được đặt trong thư mục WINDDK \ BuildNumber \ lib \ wlh.

• Include USB100.h, cũng nằm trong WINDDK \ BuildNumber \ inc \ ddk. Tiêu đề này không bắt buộc, nhưng nó chứa các khai báo cho một số macro hữu ích.

Để truy cập device, một ứng dụng phải:

1. Sử dụng interface device GUID để lấy handle đến device.

2. Sử dụng handle để khởi tạo WinUSB.

3. Sử dụng API WinUSB để định cấu hình thiết bị.

4. Sử dụng API WinUSB để liên lạc với các endpoint.

Phần này cho biết cách thực hiện các tác vụ chính này, dựa trên một ứng dụng đơn giản truy cập thiết bị Fx2 .

1. Nhận một handle cho thiết bị và Khởi tạo WinUSB

Để sử dụng API WinUSB, trước tiên bạn phải có tệp tin handle cho thiết bị và sử dụng handle đó để khởi tạo WinUSB. Hai bước đầu tiên của quy trình tương tự như các bước được sử dụng để lấy file handle cho mọi thiết bị:

1. Sử dụng interface device GUID để nhận đường dẫn device. GUID chính xác là cái mà bạn đã chỉ định trong INF được sử dụng để cài đặt WinUsb.sys.

2. Sử dụng đường dẫn device từ bước 1 để lấy file handle cho device.

3. Truyền file handle cho WinUsb\_Initialize để khởi tạo WinUSB và lấy WinUSB handle. Bạn sử dụng WinUSB handle của device để xác định device khi bạn gọi API của WinUSB functions, không phải là file handle của device.

Lấy đường dẫn device

Hàm GetdevicePath function, được hiển thị trong ví dụ sau, sử dụng interface device GUID để có được đường dẫn device. Nó tương tự như code API thiết lập được sử dụng cho hầu hết các thiết bị, nhưng được đưa vào đây để hoàn thiện. Để biết thêm thông tin, hãy xem Cài đặt API trên MSDN. Lưu ý rằng một số mã code chủ yếu là xử lý lỗi đã bị bỏ qua cho rõ ràng:

Thủ tục cơ bản như sau:

1. Nhận một handle đến thông tin của device bằng cách truyền interface device GUID mà bạn đã xác định trong INF tới SetupDiGetClassDevs. Function trả về handle HDEVINFO.

2. Gọi SetupDiEnumdeviceinterfaces để liệt kê interface device của hệ thống và lấy thông tin trên interface device của bạn. Làm như vậy:

• Khởi tạo cấu trúc SP\_device\_interface\_DATA bằng cách đặt cbSize của nó bằng với kích thước của structure.

• Vượt qua handle HDEVINFO từ bước 1, interface device GUID và tham chiếu đến SP\_device\_interface\_DATA structure lại để SetupDiEnumdeviceinterfaces.

• Khi function return, SP\_device\_interface\_DATA structure chứa dữ liệu cơ bản cho interface.

3. Gọi SetupDiGetdeviceinterfaceDetail để nhận dữ liệu chi tiết cho interface device. Thông tin được trả về trong một SP\_device\_interface\_DETAIL\_DATA structure.

• Vì kích thước của SP\_device\_interface\_DETAIL\_DATA structure khác nhau, trước tiên bạn phải có được kích thước bộ đệm chính xác bằng cách gọi SetupDiGetdeviceinterfaceDetail DeviceInterfaceDetailData

parameter set to NULL.

• function trả về kích thước bộ đệm chính xác trong tham số requiredlength. Sử dụng giá trị đó để phân bổ chính xác bộ nhớ cho một SP\_device\_interface\_DETAIL\_DATA structure.

• Gọi SetupDiGetdeviceinterfaceDetail một lần nữa và chuyển tham chiếu đến structure được khởi tạo. Khi function return, structure chứa thông tin chi tiết về interface.

4. Đường dẫn device nằm trong SP\_device\_interface\_DETAIL\_DATA structure thành viên DevicePath.

Lấy file handle cho device

Hàm Opendevice, được hiển thị trong ví dụ sau, có được file handle cho device bằng cách truyền đường dẫn device tới Createfile.

1. Gọi tiện ích GetdevicePath function để nhận đường dẫn device. GetdevicePath đã được thảo luận trong phần trước.

2. Vượt qua đường dẫn device đến Createfile để lấy file handle cho thiết bị. Ví dụ này có được một tập tin hand hỗ trợ truy cập đọc và ghi đồng bộ vào thiết bị. Để biết chi tiết về cách mở file handle cho I / O không đồng bộ, hãy xem trang tham khảo của CreateFile function trên MSDN. Đảm bảo đặt cờ file\_FLAG\_OVERLAPPED. WinUSB phụ thuộc vào cài đặt này:

Khởi tạo WinUSB

Handle đã thu được trong phần trước là file handle cho device. Tuy nhiên, API WinUSB sử dụng handle WinUSB để xác định thiết bị đích thay vì tệp xử lý tệp. Để có được một handle WinUSB , hãy khởi tạo WinUSB bằng cách chuyển file handle tới WinUsb\_Initialize. Function này trả về WinUSB handle cho device.

Ví dụ sau đây khởi tạo WinUSB với file handle mà tiện ích Opendevice function thu được, như đã thảo luận trong phần trước. Sau đó, nó lưu trữ handle WinUSB tương ứng để sử dụng sau này trong cấu trúc toàn cầu được xác định riêng tư:

Cấu hình thiết bị

Sau khi một ứng dụng khởi chạy WinUSB, nó phải cấu hình USB device. Quy trình này tương tự như quy trình mà device driver USB sử dụng. Tuy nhiên, nó được thực hiện bằng cách gọi các hàm API function của WinUSB thay vì các thư viện WDF framework hoặc bất kỳ hàm USB WDM nào của Windows.

Hàm này có tên là InitializeDevice function, được hiển thị trong ví dụ sau, cấu hình Fx2 device. Nó cũng include code được hiển thị trong ví dụ trước để khởi tạo WinUSB:

Thủ tục cơ bản như sau:

1. Nếu cần, hãy gọi WinUsb\_ QueryDeviceInformation để có được tốc độ của device. Function trả về một trong ba giá trị: LowSpeed (0x01), FullSpeed (0x02) hoặc HighSpeed (0x03).

2. Truyền handle interface device cho WinUsb\_QueryInterfaceSettings để có được descriptors interface tương ứng. Handle WinUSB tương ứng với interface đầu tiên. Vì Fx2 device chỉ hỗ trợ một interface không có alternate settings, tham số AlternateSettingNumber được đặt thành 0 và function chỉ được gọi một lần. Nếu device hỗ trợ nhiều interface, hãy gọi WinUsb\_GetAssociatedinterface để lấy interface handles cho interface được liên kết.

WinUsb\_ GetAssociatedInterface trả về USB\_interface\_DEscriptOR structure có chứa thông tin về interface. Đặc biệt, structure chứa số lượng endpoint trong interface.

3. Để có được thông tin về từng endpoint, hãy gọi WinUsb\_QueryPipe một lần cho mỗi endpoint trên mỗi interface. WinUsb\_QueryPipe trả về một cấu trúc WINUSB\_PIPE\_INFORMATION có chứa thông tin về endpoint đã chỉ định. Fx2 device có một interface với ba endpoint, do đó, tham số AlternateinterfaceNumber được đặt thành 0 và giá trị của tham số PipeIndex thay đổi từ 0 đến 2.

4. Để xác định loại pipe, hãy kiểm tra thành viên PipeInfo của WINUSB\_PIPE\_INFORMATION structure. Thành viên này được đặt thành một trong các giá trị liệt kê USBD\_PIPE\_TYPE: UsbdPipeTypeControl, UsbdPipeTypeIsochronous, UsbdPipeTypeBulk hoặc UsbdPipeTypeInterrupt.

Ba endpoint mà Fx2 device hỗ trợ là một pipe interrupt, một pipe bulk in và một pipe bulk out, vì vậy, PipeInfo sẽ được đặt thành UsbdPipeTypeInterrupt hoặc UsbdPipeTypeBulk.

Giá trị UsbdPipeTypeBulk xác định các bulk pipes , nhưng không đưa ra direction. Thông tin đó được encoded trong bit cao của địa chỉ pipe, được lưu trữ trong thành viên WINUSB\_PIPE\_INFORMATION structure. Cách đơn giản nhất để xác định direction là chuyển giá trị PipeId cho một trong các macro sau từ USB100.h:

• USB\_ENDPOINT\_ DIRECTION\_IN trả về TRUE nếu direction đi vào.

• USB\_ENDPOINT\_ DIRECTION\_OUT trả về TRUE nếu direction đi ra.

Ứng dụng sử dụng giá trị PipeId để xác định pipe mong muốn trong các cuộc gọi đến các function của WinUSB như WinUsb\_ReadPipe, vì vậy ví dụ này lưu trữ cả ba giá trị của PipeId để sử dụng sau.

Giao tiếp với các endpoint

Sau khi bạn có các giá trị PipeId cho các endpoint của device, bạn có thể giao tiếp với device. Bài viết này thảo luận về cách đưa ra các yêu cầu control, đọc và viết.

Yêu cầu control

Tất cả các thiết bị USB có các endpoint điều khiển cùng với các endpoint được liên kết với interface. Mục đích chính của endpoint control là cung cấp một endpoint mặc định mà các ứng dụng có thể sử dụng để cấu hình device. Tuy nhiên, device cũng có thể sử dụng endpoint điều khiển cho các mục đích cụ thể của thiết bị. Fx2 device sử dụng endpoint điều khiển để điều khiển thanh ánh sáng và màn hình kỹ thuật số bảy đoạn.

Các lệnh điều khiển bao gồm một gói thiết lập 8 byte, trong đó include một mã yêu cầu code chỉ định yêu cầu cụ thể và một bộ đệm dữ liệu tùy chọn. Các mã yêu cầu code và định dạng bộ đệm được xác định bởi nhà cung cấp. Để đưa ra yêu cầu control:

1. Phân bổ một bộ đệm, nếu cần thiết.

2. Xây dựng một gói thiết lập.

3. Gọi WinUsb\_ControlTransfer để gửi yêu cầu và bộ đệm đến endpoint điều khiển.

SetBar function, được hiển thị trong ví dụ sau, sẽ gửi yêu cầu điều khiển đến thiết bị Fx2 để điều khiển đèn trên thanh ánh sáng:

Code để đặt thanh ánh sáng là 0xD8, được xác định để thuận tiện là SET\_BARGRAPH\_DISPLAY. Thủ tục ban hành yêu cầu như sau:

1. Tải dữ liệu vào bộ đệm. Device yêu cầu bộ đệm dữ liệu 1 byte chỉ định phần tử nào sẽ được thắp sáng bằng cách đặt các bit thích hợp. Interface người dùng (UI) cho ứng dụng include một bộ tám điều khiển CheckBox được sử dụng để chỉ định các thành phần nào của thanh ánh sáng sẽ được thắp sáng. Ví dụ truy vấn interface người dùng để xác định đèn nào sẽ sáng và đặt các bit thích hợp trong bộ đệm.

2. Xây dựng một gói thiết lập bằng cách gán các giá trị cho một chuỗi WINUSB\_SETUP\_PACKET structure:

• Thành viên RequestType chỉ định hướng yêu cầu. Nó được đặt thành 0, cho biết truyền dữ liệu từ máy chủ đến thiết bị. Đối với chuyển thiết bị từ thiết bị đến máy chủ, hãy đặt RequestType thành 1.

• Thành viên Yêu cầu được đặt thành code do nhà cung cấp xác định cho yêu cầu này, 0xD8. Nó được định nghĩa để thuận tiện là SET\_BARGRAPH\_DISPLAY.

• Thành viên Độ dài được đặt thành kích thước của bộ đệm dữ liệu.

Các thành viên Index và Giá trị không được yêu cầu cho yêu cầu này, vì vậy chúng được đặt thành không.

3. Truyền yêu cầu bằng cách truyền handle device, WinUSB, gói thiết lập và bộ đệm dữ liệu tới WinUsb\_ControlTransfer. Function trả về số byte đã được chuyển đến device.

Yêu cầu I / O WinUSB

Fx2 device có các endpoint bulk‐in và bulk‐out có thể được sử dụng cho các yêu cầu đọc và ghi tương ứng. Hai endpoint này được định cấu hình cho loopback, vì vậy device chỉ cần di chuyển dữ liệu từ endpoint bulk‐in sang endpoint bulk‐out. Nó không thay đổi giá trị của dữ liệu hoặc tạo bất kỳ dữ liệu mới nào, do đó, yêu cầu đọc sẽ đọc dữ liệu được gửi bởi yêu cầu ghi gần đây nhất.

**Yêu cầu viết WinUSB**

WinUSB có các function riêng biệt để gửi các yêu cầu ghi và đọc: WinUsb\_WritePipe và WinUsb\_ReadPipe. WriteTodevice function, được hiển thị trong ví dụ sau, viết một chuỗi đơn giản vào thiết bị Fx2 :

Để gửi yêu cầu viết:

1. Tạo một bộ đệm và điền nó với dữ liệu mà bạn muốn ghi vào thiết bị. Mẫu thu được chuỗi Unicode 12 ký tự từ điều khiển Chỉnh sửa trên UI UI ứng dụng.

Ứng dụng chịu trách nhiệm quản lý kích thước của bộ đệm một cách thích hợp. Miễn là một ứng dụng không sử dụng RAW\_IO, không có giới hạn về bộ đệm

kích thước. WinUSB chia bộ đệm thành các phần có kích thước phù hợp, nếu cần thiết. Đối với RAW\_IO, kích thước của bộ đệm bị giới hạn bởi bộ điều khiển. Đối với các yêu cầu đọc, bộ đệm phải là bội số của kích thước gói tối đa.

2. Viết bộ đệm vào device bằng cách gọi WinUsb\_WritePipe. Vượt qua interface handle, giá trị PipeId cho pipe ra và bộ đệm. Trong trường hợp này, interface handle là handle WinUSB. Function trả về số byte thực sự được ghi vào device trong tham số byteWritten.

Tham số cuối cùng của WinUsb\_WritePipe, Overlapped, được đặt thành NULL để yêu cầu hoạt động đồng bộ. Để thực hiện ghi không đồng bộ, chồng chéo phải là một con trỏ tới cấu trúc Overlapped.

Hành vi viết WinUSB mặc định

Không có chiều dài ghi được chuyển tiếp xuống ngăn xếp. Nếu độ dài truyền lớn hơn độ dài truyền tối đa, WinUSB chia yêu cầu thành các yêu cầu nhỏ hơn có độ dài truyền tối đa và gửi chúng một cách an toàn.

Yêu cầu đọc WinUSB

Hoạt động đọc tương tự như hoạt động viết. Chuyển đến WinUsb\_ReadPipe interface handle, giá trị PipeId cho endpoint hàng loạt và bộ đệm trống có kích thước phù hợp. Khi function trở lại, bộ đệm chứa dữ liệu được đọc từ thiết bị. Số lượng byte đã được đọc được trả về trong tham số byteRead của function. Ví dụ sau đây đọc một chuỗi ký tự từ thiết bị Fx2 và hiển thị nó trong điều khiển static trên UI UI của ứng dụng:

WinUSB mặc định Hành vi đọc

Không có độ dài đọc hoàn thành ngay lập tức với thành công và không được gửi xuống ngăn xếp. Nếu độ dài truyền lớn hơn độ dài truyền tối đa, WinUSB chia yêu cầu thành các yêu cầu nhỏ hơn có độ dài truyền tối đa và gửi chúng một cách an toàn. Nếu độ dài truyền tải không phải là bội số của endpoint Max MaxPquetSize, WinUSB sẽ tăng kích thước của chuyển khoản lên bội số tiếp theo của MaxPacketSize. Nếu device trả về nhiều dữ liệu hơn yêu cầu, WinUSB sẽ lưu dữ liệu thừa. Nếu dữ liệu remain từ lần đọc trước, WinUSB sao chép nó vào đầu của lần đọc tiếp theo và hoàn thành việc đọc, nếu cần.

Chính sách ống

WinUSB cho phép sửa đổi hành vi mặc định của nó thông qua các chính sách có thể được áp dụng cho pipe (endpoint). Sử dụng các chính sách có thể giúp các IHV điều chỉnh WinUSB để phù hợp nhất với thiết bị của họ với khả năng của nó. Bảng 4 cung cấp danh sách các chính sách pipe mà WinUSB hỗ trợ.

Bảng 4. Chính sách ống WinUSB

Số chính sách Tên chính sách Mặc định

0x01 SHORT\_PACKET\_TERMINATE Tắt

0x02 AUTO\_CLEAR\_STALL Tắt

0x03 PIPE\_TRANSFER\_TIMEOUT 5 giây cho điều khiển, 0 cho người khác

0x04 IGNORE\_SHORT\_PACKETS Tắt

0x05 ALLOW\_PARTIAL\_READS Bật

0x06 AUTO\_FLUSH Tắt

0x07 RAW\_IO Tắt

0x09 RESET\_PIPE\_ON\_RESUME Tắt

Bảng 5 cung cấp thông tin về thời điểm bạn nên sử dụng từng chính sách pipe và mô tả hành vi kết quả khi mỗi chính sách được bật.

Bảng 5. Hành vi chính sách của ống

Chính sách Khi nào nên sử dụng Hành vi

SHORT\_PACKET\_TERMINATE § Sử dụng nếu thiết bị của bạn yêu cầu các lần chuyển OUT bị chấm dứt với một gói ngắn (hầu hết các thiết bị không yêu cầu điều này).

Policy Chính sách này chỉ hợp lệ đối với các endpoint OUT hàng loạt và gián đoạn (đặt chính sách này thành IN

endpoint không có hiệu lực). Nếu được bật, tất cả ghi vào endpoint sẽ được kết thúc bằng một gói ngắn.

AUTO\_CLEAR\_STALL Sử dụng nếu bạn muốn chuyển không thành công không để endpoint ở trạng thái bị đình trệ.

Policy Chính sách này chỉ hợp lệ đối với số lượng lớn và ngắt IN endpoint.

Policy Chính sách này thường chỉ hữu ích khi bạn chuyển nhiều lần đọc đến endpoint khi RAW\_IO bị tắt. § Nếu được bật, khi chuyển IN không thành công và trả về trạng thái không phải là STATUS\_CANCELLED hoặc STATUS\_device\_NOT\_CONNECTED, WinUSB sẽ đặt lại endpoint trước khi hoàn thành chuyển không thành công.

Nếu bị tắt, các lần chuyển tiếp theo đến endpoint sẽ thất bại cho đến khi endpoint được đặt lại theo cách thủ công bằng cách gọi WinUsb\_ResetPipe ().

Không có hiệu suất đáng kể

sự khác biệt tồn tại giữa việc kích hoạt hoặc vô hiệu hóa chính sách này.

Chính sách Khi nào nên sử dụng Hành vi

PIPE\_TRANSFER\_TIMEOUT Sử dụng nếu bạn có endpoint mà bạn muốn chuyển hoàn thành trong một thời gian cụ thể. Nếu được đặt thành 0, chuyển sẽ vô thời hạn cho đến khi chúng bị hủy theo cách thủ công hoặc chúng được hoàn thành bình thường.

Nếu được đặt thành giá trị khác không, bộ hẹn giờ sẽ bắt đầu khi yêu cầu được gửi xuống ngăn xếp USB lõi (yêu cầu không hết thời gian trong khi xếp hàng WinUSB). Khi hết giờ, yêu cầu bị hủy.

Có một hiệu suất nhỏ

phạt vì quản lý bộ hẹn giờ.

IGNORE\_SHORT\_PACKETS § Sử dụng nếu bạn không muốn các gói ngắn hoàn thành các yêu cầu đọc của mình (điều này rất hiếm).

Policy Chính sách này chỉ hợp lệ đối với hàng loạt và ngắt IN endpoint với RAW\_IO bị vô hiệu hóa. Chỉ đọc hoàn thành nếu xảy ra lỗi, yêu cầu bị hủy hoặc tất cả các byte được yêu cầu đã được nhận.

ALLOW\_PARTIAL\_READS § Sử dụng nếu thiết bị của bạn có thể gửi nhiều dữ liệu hơn so với yêu cầu (chỉ có thể nếu bạn đang yêu cầu không phải là đa dạng của MaxPacketSize).

Sử dụng nếu ứng dụng muốn đọc một vài byte để xác định tổng số byte cần đọc.

Policy Chính sách này chỉ hợp lệ đối với số lượng lớn và ngắt IN endpoint. Nếu bị tắt và device trả về nhiều dữ liệu hơn yêu cầu, yêu cầu sẽ hoàn thành với một lỗi.

Nếu được bật, WinUSB ngay lập tức hoàn thành các yêu cầu đọc cho byte không thành công và không gửi yêu cầu xuống ngăn xếp.

Nếu được bật và device trả về nhiều dữ liệu hơn yêu cầu, WinUSB sẽ lưu dữ liệu dư thừa và đưa nó vào lần đọc tiếp theo

yêu cầu.

AUTO\_FLUSH Sử dụng nếu thiết bị của bạn có thể gửi nhiều dữ liệu hơn yêu cầu và ứng dụng không yêu cầu bất kỳ dữ liệu bổ sung nào ngoài những gì được yêu cầu (chỉ có thể nếu yêu cầu không bội số MaxPack etSize).

Policy Chính sách này chỉ hợp lệ đối với hàng loạt và ngắt trong các endpoint trong đó ALLOW\_PARTIAL

Chính sách \_READS được bật. Nếu endpoint trả về nhiều byte hơn yêu cầu, WinUSB sẽ loại bỏ các byte đó mà không gặp lỗi.

Không có hiệu lực nếu ALLOW\_PARTIAL\_READS bị tắt.

Chính sách Khi nào nên sử dụng Hành vi

RAW\_IO Sử dụng nếu hiệu suất đọc là ưu tiên và ứng dụng gửi nhiều lần đọc đồng thời đến cùng một endpoint.

Policy Chính sách này chỉ hợp lệ đối với các endpoint IN hàng loạt và ngắt (thiết lập chính sách trên các endpoint OUT không có hiệu lực). Yêu cầu đến endpoint không phải là bội số của MaxPacketSize.

Yêu cầu đến endpoint lớn hơn kích thước truyền tối đa (có thể truy xuất được thông qua chính sách pipe MAXIMUM\_TRANSFER\_SIZE) không thành công.

Tất cả các lần đọc được định dạng tốt đến endpoint sẽ ngay lập tức được gửi xuống ngăn xếp để được lên lịch trong bộ điều khiển máy chủ.

Cải thiện hiệu suất đáng kể xảy ra khi bạn gửi nhiều yêu cầu đọc.

Độ trễ giữa gói cuối cùng

của một lần chuyển và gói đầu tiên của lần chuyển tiếp theo bị giảm.

RESET\_PIPE\_ON\_RESUME § Sử dụng nếu thiết bị không bảo vệ trạng thái chuyển đổi dữ liệu của nó qua trạng thái treo.

Policy Chính sách này chỉ hợp lệ cho các endpoint hàng loạt và ngắt. Khi tiếp tục tạm dừng, WinUSB đặt lại endpoint trước khi cho phép các yêu cầu được gửi đến endpoint.

Để biết thêm thông tin về các chính sách của WinUSB, hãy xem Hướng dẫn WinUsb\_GetPipePolicy và và Win WinUsb\_SetPipePolicy, trong WDK.

Quản lý năng lượng WinUSB

WinUSB sử dụng các máy trạng thái KMDF để quản lý năng lượng. Chính sách quyền lực được quản lý thông qua các cuộc gọi đến WinUsb\_SetPowerPolicy, cũng như các mặc định được đặt trong Registry. Để cho phép device đánh thức hệ thống, bạn phải thêm giá trị SystemWakeEnables DWORD Registry và đặt nó thành giá trị khác không. Một hộp kiểm trong trang Thuộc tính device được bật tự động để người dùng có thể ghi đè cài đặt.

Đình chỉ chọn lọc

Đình chỉ chọn lọc có thể bị vô hiệu hóa bởi bất kỳ cài đặt hệ thống hoặc WinUSB nào. Một cài đặt không thể buộc WinUSB kích hoạt tạm dừng chọn lọc. Các cài đặt Chính sách nguồn sau đây ảnh hưởng đến hành vi của đình chỉ chọn lọc:

AUTO\_SUSPEND

Khi được đặt thành 0, không treo có chọn lọc thiết bị. TẠM NGỪNG

Đặt thời gian giữa device không hoạt động và WinUSB yêu cầu

device để đình chỉ chọn lọc.

Các key của Registry sau đây ảnh hưởng đến hành vi của đình chỉ chọn lọc:

deviceIdleEnables

Khi được đặt thành 0, không treo có chọn lọc thiết bị.

deviceIdleIgnoreWakeEnable

Khi được đặt thành giá trị khác 0, tạm dừng thiết bị ngay cả khi không hỗ trợ

Điều khiển từ xa.

Người dùng SetdeviceIdleEnables

Khi được đặt thành giá trị khác 0, hãy bật hộp kiểm trong trang Thuộc tính device cho phép người dùng ghi đè mặc định không hoạt động.

Mặc địnhIdleState

Đặt giá trị mặc định của cài đặt chính sách nguồn AUTO\_SUSPEND. Key Registry này được sử dụng để bật hoặc tắt tạm dừng có chọn lọc khi một handle không mở cho thiết bị.

Mặc địnhIdleTimeout

Đặt trạng thái mặc định của cài đặt chính sách năng lượng SUSPEND\_DELAY.

Phát hiện nhàn rỗi

Tất cả các chức năng ghi và điều khiển chuyển thiết bị buộc thiết bị vào trạng thái nguồn D0 và đặt lại bộ hẹn giờ không hoạt động. Các hàng đợi endpoint IN không được quản lý năng lượng. Đọc đánh thức thiết bị khi chúng được gửi. Tuy nhiên, một thiết bị có thể trở nên nhàn rỗi trong khi yêu cầu đọc chờ xử lý.

Cân nhắc tính năng trong tương lai

Vui lòng gửi yêu cầu của bạn cho các tiện ích mở rộng và tính năng bổ sung tới [WinUSBFB@microsoft.com](mailto:WinUSBFB@microsoft.com). Một số tính năng có thể có trong tương lai include:

• Hỗ trợ endpoint đồng bộ.

• Tiện ích mở rộng tính năng USB3.

Tóm lược

Bằng cách sử dụng WinUSB, IHV cung cấp giải pháp driver Windows vững chắc cho thiết bị phần cứng USB của họ. WinUSB hỗ trợ Windows XP và các phiên bản Windows mới hơn và hỗ trợ cả phiên bản Windows 32 bit và 64 bit. Các ứng dụng Windows 32 bit và 64 bit hiện có có thể dễ dàng sửa đổi để tận dụng API WinUSB. WinUSB loại bỏ tất cả các vấn đề liên quan đến driver và cho phép IHV cung cấp giải pháp Windows driver cho thiết bị USB của họ trong thời gian ngắn hơn và tốn ít công sức hơn so với việc viết driver Windows của riêng họ.

Tài nguyên

Trung tâm nhà phát triển phần cứng Windows (WHDC):

kernel ‐ Chế độ code Đăng nhập Walkential

<http://www.microsoft.com/whdc/winlogo/drvsign/kmcs_walk> phiên.mspx

Windows driver Foundation (WDF)

<http://www.microsoft.com/whdc/do/wdf/default.mspx>

Viết driver USB với WDF

<http://www.microsoft.com/whdc/do/wdf/USB_WDF.mspx>

Bộ điều khiển Windows:

Cài đặt device và driver

<http://go.microsoft.com/fwlink/?LinkId=98295>

Hướng dẫn thiết kế cài đặt device và driver

<http://go.microsoft.com/fwlink/?LinkId=146855>

Kết thúc Cài đặt hành động (Windows Vista trở lên)

<http://go.microsoft.com/fwlink/?LinkId=146856>

Kết thúc ‐ Cài đặt trang Wizard

<http://go.microsoft.com/fwlink/?LinkId=146857>

Các lớp USB được hỗ trợ

<http://go.microsoft.com/fwlink/?LinkId=98297>

USB

<http://go.microsoft.com/fwlink/?LinkId=98298>

WinUSB

<http://go.microsoft.com/fwlink/?LinkId=98299>

Người dùng WinUSB ‐ Chế độ Hỗ trợ khách hàng Routs

<http://go.microsoft.com/fwlink/?LinkId=98300>

WinUsb\_GetPipePolicy

<http://go.microsoft.com/fwlink/?LinkId=146858>

WinUsb\_SetPipePolicy

<http://go.microsoft.com/fwlink/?LinkId=146859>

MSDN:

Createfile function

<http://go.microsoft.com/fwlink/?LinkId=98294>

API thiết lập

<http://go.microsoft.com/fwlink/?LinkId=98296>

Báo chí của Microsoft:

Phát triển driver với Windows driver Foundation

<http://www.microsoft.com/MSPress/books/10512.aspx>