Firmware para dispositivo esclavo USB de Clase HID

Emanuel G. Aguirre, Pablo A. Di Giulio

Universidad Tecnológica Nacional, Facultad Regional San Francisco

Abstract

El objetivo del presente trabajo es desarrollar de la capa de Software (Firmware) del Protocolo USB para un dispositivo Esclavo USB de Clase HID.

Un dispositivo Esclavo USB provee alguna función al Host USB; un Esclavo es un dispositivo tal como un mouse, un teclado, un joystick, una impresora, etc.

Se denomina Clase, al conjunto de Dispositivos USB que proveen al Host funciones similares. Los dispositivos de la Clase HID (Human Interface Devices) son aquellos que tienen como objetivo permitir la interacción entre el Host (PC, palm, etc.) y el usuario, de ahí su nombre; entre dichos dispositivos, pueden darse como ejemplo: mouse, teclado, joystick, keypad.

Ya que el Protocolo USB está compuesto por capas de Hardware y de Software, dentro una Clase particular de dispositivos, el Hardware es similar o igual, y lo que varía de una función a otra es el Software. Entre diferentes tipos de capas de Hardware varía la velocidad de transmisión de datos (low-speed, full-speed y high-speed), pero el formato en que se envían los datos y el protocolo es el mismo.

En el presente trabajo, se desarrolló la capa de Software correspondiente al Protocolo USB utilizado por la Clase HID. La capa de Hardware se encuentra embebida en un Microcontrolador (MCU).

La capa de Software para la implementación del Protocolo USB para un Esclavo Clase HID se desarrolló en lenguaje C sobre la plataforma de hardware elegida (MCU Freescale 68HC908JB8) y luego se aplicó a las subclases Mouse y Keyboard.

Palabras Clave

Arquitectura de Computadoras. Comunicaciones.

1. Introducción

El Bus USB por Universal Serial Bus (en adelante USB), fue ideado para ser un bus de extensión para PC, con el objetivo de llegar a ser un estándar de la industria. Las posibles tasas de transferencia son: 1,5 Mbits/s, 12 Mbits/s y 480 Mbits/s, lo cual lo hace útil para aplicaciones que van desde periféricos de PC hasta dispositivos de video u otros que requieran alta tasa de transferencia de datos.

Otro motivo fue la creación de un bus que fuera independiente de la plataforma de hardware utilizada, al hacer que gran parte del funcionamiento dependa del software.

Otro motivo es la creación de una arquitectura de bus que permitiera conectar los periféricos de la PC con un mismo conector y que además fuera Hot Plug And Play, haciéndolo de esta forma más simple para el usuario; evitando así la multiplicidad de conectores tales como el Puerto Serial, Puerto Paralelo, PS/2, Gameport, y demás derivados del diseño original de la PC de IBM de los años 80.

Aún otro motivo es la reducción de costo. Dándose ésta en la reducción de cables de cobre (por ser un bus de tipo serial), en la eliminación de la amplia variedad de conectores (siendo todos reemplazados por un solo tipo).

La alternativa actual al USB es el bus FireWire (IEEE-1394), creado por Apple Computer. El bus IEEE-1394 es más rápido y más flexible que el USB, pero es más caro. La ventaja del USB es que es más útil para periféricos de baja tasa de transferencia de datos, tales como periféricos de PC. Mientras que en el USB el Host es el que controla las comunicaciones, el IEEE-1394 utiliza el modelo peer-to-peer en el cual los periféricos pueden comunicarse entre sí. El IEEE-1394 tiene una taza de transferencia de 400 Mbits/s. Los 480 Mbits/s del USB

2.0 full-speed son actualmente superados por el IEEE-1394b que llega a los 3,6 Gb/s.

La motivación principal para la realización de este trabajo fue profundizar en esta tecnología, que actualmente es muy usada en el mundo, pero que en Argentina se utiliza en forma escasa.

El objetivo del trabajo fue obtener el conocimiento y las aptitudes suficientes para lograr la utilización del bus USB como interfaz con una PC para aplicaciones de uso general. Luego se procedió a realizar la comunicación con el Driver del Host (PC) de la Clase HID del sistema operativo. El Driver del Host de la Clase HID está disponible en todos los sistemas operativos de uso común en PCs.

La Clase HID, es una subclase de dispositivos USB cuya función es proveer a la PC de una interfaz con el usuario (mouse, teclado, joystick, etc.). Está especificada como un estándar, definido por el USB Implementers Forum y su versión actual es la HID 1.11.

2. Elementos del Trabajo y metodología

Los elementos utilizados para el presente trabajo fueron los siguientes:

-Placa de desarrollo con un Microcontrolador (MCU) Freescale 68HC908JB8 (Hardware).

-Freescale CodeWarrior Development Studio v5.7 para MCUs 68HC(S)908 (Compilador).

-Snoopy Pro v0.22 para Windows XP (USB Sniffer).

-Perisoft Bus Hound v6.0 para Windows XP (USB Sniffer).

En la arquitectura USB las capas de hardware y de software tienen a cargo distintas funciones, tanto en el Esclavo como en el Host.

El MCU 68HC908JB8 se usó para implementar la capa de hardware del Esclavo USB. El IDE CodeWarrior se utilizó para programar la capa de software del Esclavo USB, simularla, y para grabarla en la memoria Flash del MCU. Los USB Sniffers se utilizaron para analizar las estructuras de datos del sistema operativo que reflejan el tráfico en el bus USB y así depurar el Firmware del MCU.

2.1. Arquitectura del bus USB

Para lograr un mínimo grado de comprensión del presente trabajo, es necesario analizar los elementos del bus USB desde el punto de vista de los sistemas de comunicaciones.

2.1.1. Elementos del USB

El USB es un bus ideado para intercambio de datos entre un computador anfitrión (Host), y dispositivos conectados a él (Esclavos). Los periféricos conectados al USB comparten el ancho de banda del bus mediante un protocolo basado en mensajes (tokens).

Un sistema USB consta de 3 partes:
- Anfitrión USB (USB Host o Host).
- Dispositivos USB (USB devices).
- Interconexión USB (USB interconnect).

Existe sólo un Host en un sistema USB. Los dispositivos USB proveen servicios o funciones al Host. La interconexión USB es la que soporta el tráfico entre el Host y los dispositivos USB, es el canal da comunicación.

2.1.2. Topología

La topología física del USB es de tipo estrella jerarquizada. Con un máximo de 7 niveles de jerarquía.

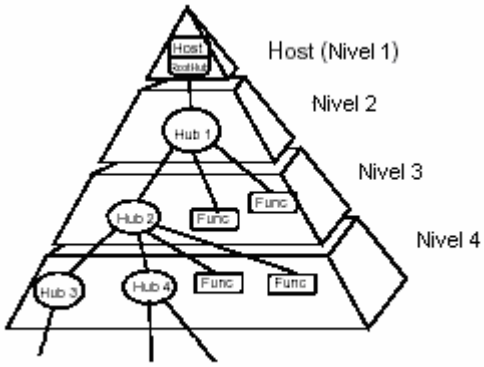

Fig. 1. Topología Física del USB

La topología lógica del USB es de tipo estrella. Lo que implica que en un dispositivo físico puede haber

implementado más de un dispositivo lógico (por ej.: un teclado con un mouse incluído).

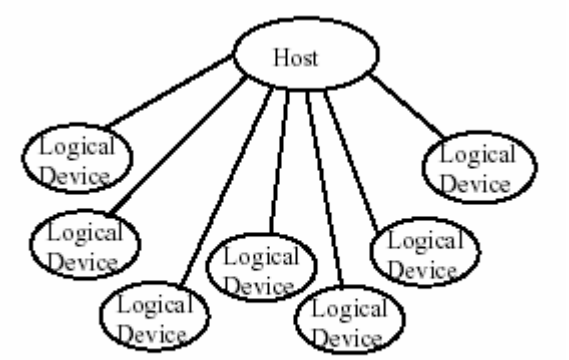

Fig. 2. Topología lógica del USB

2.1.3. Direccionamiento

El direccionamiento de dispositivos Esclavos USB se hace mediante un número de 7 bits, lo que hace posible direccionar hasta 128 dispositivos. En el caso de utilizar HUBs, en teoría, pueden conectarse hasta 127 dispositivos USB por HUB.

Dentro de cada dispositivo lógico USB existen uno o más Endpoints, que son receptores independientes de datos.

El ruteo de los datos es de tipo Broadcast. Esto es, todos los esclavos dentro de una jerarquía reciben los datos, pero solo los procesa el Esclavo al que están destinados dichos datos.

2.1.4. Acceso al canal de comunicación

El protocolo de acceso al canal de comunicación (media access protocol) es de tipo polled. Por lo tanto, cuando el Host se comunica con un Esclavo, lo hace mediante el envío de un token con la dirección de dicho Esclavo al canal de comunicación (bus), solo el Esclavo con esa dirección procesa el pedido del Host.

2.1.5. Tasas de transferencia de datos

Las posibles tasas de transferencia de datos para el USB son: 1,5 Mbits/s, 12 Mbits/s y 480 Mbits/s. La correspondiente denominación de cada una de ellas es:

- low-speed: 1,5 Mbits/s,
- full-speed: 12 Mbits/s,
- high-speed: 480 Mbits/s.

Cada una de las tasas de transferencia anteriores tiene ciertas características propias de funcionamiento y configuración.

2.1.6. Transmisión y codificación

Los datos son transmitidos en forma serie, en 2 líneas de datos complementarias denominadas D+ y D-. Además se proveen 2 líneas de alimentación y de masa respectivamente, las cuales pueden servir para que el dispositivo tome alimentación del Host (5 V, 500 mA máx.).

Para transmitir los datos en forma serie se utiliza la codificación Non-Return-To-Zero-Inverted o NRZI. En este tipo de codificación, un 0 (cero) se representa sin un cambio de nivel en la tensión, y un 1 (uno) se representa con un cambio de nivel en la tensión. Conjuntamente, se utiliza el bit stuffing, técnica que consiste en insertar un 0 (cero) cada 6 (seis) 1s (unos) consecutivos en el flujo de bits. Además, del bit stuffing y de la codificación NRZI, se utilizan CRCs. Los CRCs se generan después del bit stuffing.

2.2. Protocolo USB

En el USB, lo datos se envían en paquetes. A su vez, los Paquetes se agrupan para formar las Transacciones, y las Transacciones se agrupan para formar las Transferencias. Las Transferencias son las estructuras de datos que tienen sentido para el Software Client que corre en el Host, y que es el destinatario final de los datos enviados a o recibidos desde el dispositivo lógico.

2.2.1. Endpoints, Buffers y Pipes

Cada canal lógico de comunicación entre el Host y el Esclavo está formado por una tríada Endpoint-Pipe-Buffer. El Endpoint pertenece al Esclavo USB, el Buffer pertenece al Host, y el Pipe es la conexión lógica entre ambos.

Existen 2 tipos básicos de Endpoints: Los de Control y los de Datos. Los de Control son utilizados para transferir información de configuración y estado entre Esclavo y el Host. Los de Datos son

utilizados para transferir datos que utiliza el software que corre en el Host.

Cada Endpoint Está identificado por un número.

Los Endpoints de Control pueden transferir datos en ambas direcciones.

La clasificación de los Endpoints de datos según la dirección en la que transfieren los datos es:

- IN: Los datos van del Esclavo al Host.
- OUT: Los datos van del Host al Esclavo.

Los Endpoints se agrupan para formar Interfaces. Una Interface representa a un dispositivo lógico USB.

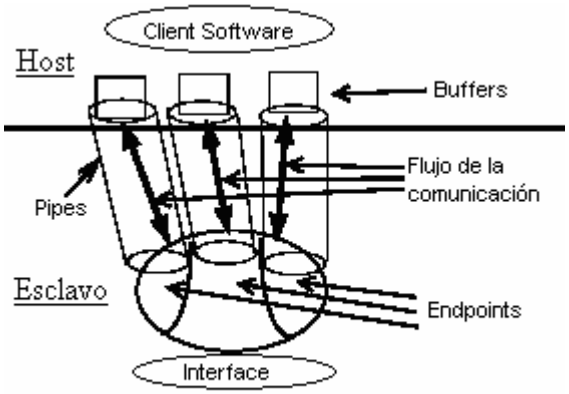

Fig. 3. Endpoints, Pipes y Buffers

Cada tipo de Endpoint está asociado a un tipo de Transferencia.

2.2.2. Transferencias

Una Transferencia es un bloque de datos que conforma una estructura comprensible para el Host o el Esclavo.

Existen 2 tipos básicos de Transferencias, que estén directamente relacionadas a los tipos de Endpoints: las de Control y las de Datos.

Dentro de las Transferencias de datos existen 3 tipos:

- Transferencias de Interrupción (Interrupt Data Transfers): Baja tasa de transferencia de datos, es una reliable data transfer.
- Transferencias de Bultos (Bulk Data Transfers): Para transferir cantidades relativamente grandes de datos en un solo envío. El ancho de banda disponible para este tipo de transferencia varía en función de la disponibilidad de éste.
- Transferencias Isócronas (Isochronous Data Transfers): Ocupan un ancho de banda del USB, cuya latencia es negociada en la configuración. Sirven para transmitir datos a intervalos regulares de tiempo, es una unreliable data transfer.

Cada Endpoint de un dispositivo USB está asociado a un y sólo un tipo de Transferencia. El Endpoint0 siempre realiza Control Transfers y todo los Esclavos USB lo tienen. A la Pipe asociada al Endpoint0 se la denomina Default Control Pipe.

El Host obtiene la información sobre el tipo de transferencia que realizará cada Endpoint durante el proceso de enumeración, cuando lee los Descriptores.

Las Transferencias de Control siempre inician con un paquete SETUP, el cual tiene información sobre la función de configuración que el Host desee que el Esclavo realice. Las Transferencias de datos siempre comienzan con paquetes IN u OUT.

2.3. Comportamiento de Esclavos USB

Un Esclavo USB tiene una cantidad finita de estados posibles.

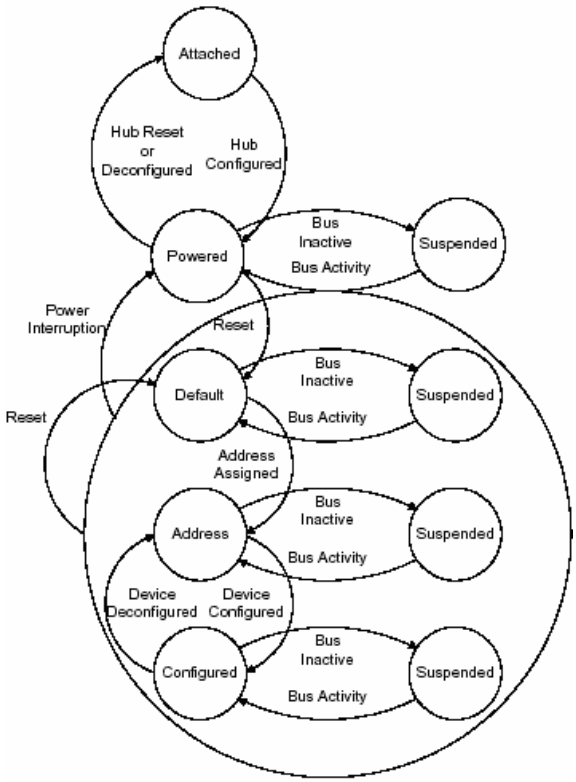

Fig. 4. Diagrama de Estados de un Esclavo USB

A continuación se describe cada uno de los posibles estados de un Esclavo USB:
- ATTACHED: Es el primer estado en que se encuentra el Esclavo apenas es conectado físicamente al bus USB.
- POWERED: Se considera que un Esclavo USB está en este estado, cuando se ha aplicado Vbus al dispositivo, o cuando se ha alimentado con una fuente externa.
- DEFAULT: Un Esclavo USB está en este estado después de que ha recibido una señal de Reset USB. Después de resetearse, éste puede comunicarse con el Host mediante el Endpoint0. Cuando el proceso de reset ha sido completado con éxito, el dispositivo es capaz de comunicarse a la velocidad correcta. La selección para low-speed y full-speed se hace con resistores de terminación. Después de haber sido reseteado, el dispositivo debe ser capaz de responder correctamente y devolver al host los Device Descriptor y Configuration Descriptor, que contienen la información sobre el Esclavo.
- ADDRESSED: Un Esclavo USB se encuentra en este estado una vez que le ha sido asignada una dirección USB por el Host.
- CONFIGURED: Antes de que el Esclavo USB pueda prestar su utilidad, éste debe se configurado. Desde el punto de vista del dispositivo, éste estará configurado una vez que haya podido procesar correctamente un Request del tipo SetConfiguration().
- SUSPENDED: Este estado es usado para ahorrar energía. Un Esclavo entra en este estado cuando no ha recibido un paquete EOP durante 3 ms. Un dispositivo debe salir de este estado cuando detecta actividad en el bus. Durante este estado se mantiene la dirección USB que fue asignada al dispositivo USB.

2.3.1. Enumeración

Se denomina Enumeración (bus enumeration), al proceso, por el cual el Host identifica a un dispositivo USB, leyendo sus Descriptores; y luego le asigna una dirección USB. Además de detectar cuando un dispositivo es removido del bus, liberar la dirección USB que este tenía asignada, y liberar los demás procesos asociados a las Pipes creadas para comunicarse con dicho dispositivo.

2.3.2. USB Device Requests

Todos los dispositivos USB responden a los Requests del Host mediante la Default Control Pipe. Los Requests se manifiestan en las llamadas Control Transfers (Transferencias de Control). Una Transferencia de Control se reconoce porque comienza con un paquete de tipo SETUP; a diferencia de los otros tipos de Transferencias que comienzan siempre con paquetes tipo IN u OUT.

Los Requests se dividen en:
- Standard Device Requests: Son los comunes a todos los tipos de dispositivos USB. Sirven para que el Host identifique y configure un dispositivo durante el proceso de Enumeración.
- Class Requests: Son los relativos a cualquier clase particular de dispositivo USB. Cada clase de dispositivo (excepto las clases genéricas o definidas por los fabricantes, vendor specific) está definida en una especificación de clase USB. Cada clase tiene sus requisitos propios, tales como tipo y número mínimo de Endpoints; además de otro tipo de Descriptor, que es llamado Report Descriptor (Descriptor de Reporte), el cual indica al Host cómo deben interpretarse los datos que envía el Esclavo al Host según la función del Esclavo.

2.3.3. Descriptores

Los Descriptores son tablas en las que los Esclavos almacenan información sobre sus características. Dichas tablas son no modificables por el Host (grabadas en ROM). Los Descriptores son jerárquicos. Algunos pueden contener información relativa a String Descriptors, que tienen información para que el Host muestre al usuario.

El Host solicita los Descriptores a los Esclavos USB mediante las Control Transfers. Los Descriptores que están en los niveles más altos de la jerarquía de

Descriptores, informan al Host sobre la existencia de los Descriptores que están en los niveles más bajos de la jerarquía de Descriptores.

A continuación se describen los Descriptores comunes a todos los tipos de Esclavos USB:

- Device Descriptor: Contiene información sobre el máximo tamaño de paquete que soporta el Endpoint0, cuántas configuraciones soporta el Esclavo, y otra información. Es el primero que lee el Host.
- Configuration Descriptor: Existe uno por cada posible forma de operar del Esclavo (solo puede haber una configuración activa en un determinado momento). Tiene información sobre cuántas Interfaces existen por configuración.
- Interface Descriptor: Tiene información sobre el número de Endpoints (excepto el Endpoint0) que utiliza al Interface y sobre la Clase a la que pertenece.
- Endpoint Descriptor: Existe uno por cada Endpoint de una Interface. Tiene información sobre el número de Endpoint. También sobre el tipo de Transferencia que realiza (Control, Interrupt, Bulk o Isochronous). Y también tiene información sobre el tamaño máximo de paquete del Endpoint.

A continuación se describen los Descriptores básicos de las Clases USB:

- Class Descriptor: Especifica a qué Clase USB pertenece una Interface. También da información sobre la longitud del Report Descriptor.
- Report Descriptor: Tiene información sobre cómo debe el Host interpretar las Transferencias del Esclavo. Su estructura es totalmente diferente al del resto de los Descriptores. En el contexto del USB, una Transferencia es equivalente a un Report, y es un bloque de datos que el Host puede interpretar.

2.4. Modelo de capas del USB

El USB tiene su modelo particular de capas. En el cual existen 3 de ellas. Por lo general, las 2 capas superiores se implementan en software y la inferior en hardware, tanto en el Esclavo como en el Host.

La Function Layer (Capa de Función) está formada por las Interfaces. Cada Interface es está formada por un grupo de Endpoints. Los datos que se mueven no tienen formato USB, tienen la estructura definida en el Report Descriptor; son los datos que corresponden al par Client Software-Function.

En la USB Device Layer (Capa de Dispositivo USB), la información transmitida entre los pares son las Transacciones; que son entre Endpoints y Buffers.(estos últimos, implementados en software en el Host).

En la USB Bus Interface Layer, los datos intercambiados entre los pares son Transacciones. Estas Transacciones están encuadradas dentro de los Frames generados a intervalos regulares de tiempo por el Host Controller.

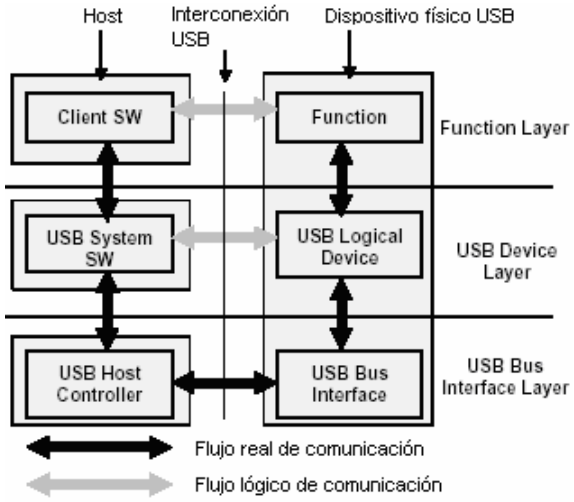

Fig. 5. Modelo de capas del USB

2.5. Clase HID

El nombre HID es la abreviatura de "Human Interface Devices". Esta Clase, cuya versión actual es el estándar HID 1.11 fue ideada con el propósito de englobar a dispositivos que permitan la interacción del usuario (ser humano) con el Host. Por lo tanto, los requerimientos de ancho de banda son mínimos, y la transferencia de datos debe ser confiable.

Los datos que los dispositivos HID envían al Host son interpretados por el HID Class Driver del sistema operativo, para luego poder ser utilizados por la aplicación que los requiera (Client Software).

Los dispositivos HID se comunican con el HID Class Driver mediante la (Default) Control Pipe o mediante Interrupt Pipes.

Los requisitos para la implementación de un dispositivo HID son:
- Control Endpoint (Endpoint0): obligatorio
- Interrupt IN Endpoint: obligatorio
- Interrupt OUT Endpoint: opcional

2.6. Hardware utilizado

Para la implementación del Esclavo USB se utilizó un MCU Freescale (ex Motorola) 68HC908JB8 de 8 bits.

El MCU utilizado posee un módulo de Interfaz de bus USB (Esclavo) cuyos registros de datos y control están mapeados en la memoria del MCU.

El módulo USB embebido en el MCU posee 3 Endpoints. El máximo payload de los Endpoints de este dispositivo es de 8 bytes. El Endpoint0 es de Control. El Endpoint1 es de tipo Interrupt IN. El Endpoint2 puede configurarse para funcionar como Interrupt IN o como Interrupt OUT. Todos los Endpoints del MCU utilizado son low-speed, por lo que tienen una tasa de transferencia de 1,5 Mbits/s; por lo que la máxima tasa de transferencia de datos útiles es de 64 Kbits/s.

El módulo USB del MCU se configuró para trabajar de modo que produzca una Interrupción cada vez que se produzca un evento de envío o recepción en el bus USB.

2.7. Firmware desarrollado

El firmware del MCU en que se implementó el Esclavo USB HID se escribió en lenguaje C.

El módulo USB del MCU se encarga de generar y decodificar los paquetes adicionales a los datos, los CRCs, el bit stuffing y la codificación NRZI. Por lo que el firmware debe encargarse de interactuar con los Drivers del sistema operativo.

Para que el Esclavo USB funcione correctamente, el firmware del MCU debe cumplir con los siguientes:
- Contener los Descriptores.
- Implementar los Estados posibles de un Esclavo USB (Finite State Machine).
- Procesar los Standard Requests.
- Procesar los Class Requests.
- Enviar los Reports de acuerdo a lo definido en su(s) Report(s) Descriptor(s).

2.7.1. Implementación de Descriptores

Cada tipo de Descriptor se declaró como una estructura “const struct”, y luego se inicializaron con sus correspondientes valores, excepto el Report Descriptor, que se declaró como un arreglo de caracteres.

Ya que cuando el Host solicita el Configuration Descriptor, el Esclavo debe enviar este y todos sus Descriptores subordinados en forma concatenada, se creó una estructura que los contuviera a todos; y es esta estructura la que se devuelve cuando se solicita el Configuration Descriptor.

2.7.2. Implementación de Standard Requests

Cada Standard Request se implementó como una función separada en el firmware.

Para identificar el tipo de Standard Request enviado por el Host, se leen los campos de bits del paquete SETUP de la Control Transfer que contiene al Request.

La determinación de los parámetros del Standard Request solicitado también se hace en base a una variable que forma parte del SETUP Packet. Como ejemplo, puede citarse que al Standard Request GetDescriptor() le corresponde un parámetro que es el Descriptor solicitado. Los parámetros son dependientes del Request.

Existe un flag en los registros de control del módulo USB del Esclavo, que indica cuando un paquete es de SETUP.

2.7.3. Implementación de Class Requests

La decodificación de los Class Requests se hace con el mismo método que la decodificación de los Standard Requests. Con la única salvedad que los Class Requests y sus parámetros son con referencia a la Clase USB implementada.

2.7.4. Reports

Todos los Reports (enviados al Host cuando este los solicita mediante el Endpoint correspondiente) deben tener siempre el tamaño y estructura declarados en el Report Descriptor, los cuales dependen de la función a implementar en el Esclavo; en caso de no cumplirse esto, el Host los recibirá, pero los descartará.

2.7.5. Otras consideraciones del firmware

El firmware debe, además, ser capaz de procesar un Bus Reset, el cual es una señal enviada por el Host, y que es señalada por un flag en un registro de estado del MCU.

3. Resultados

Los resultados del trabajo se obtuvieron trabajando sobre una PC Host cuyo sistema operativo era Microsoft Windows XP, la funcionalidad que presta el firmware implementado funciona de igual forma independientemente del Host al que se conecte el dispositivo, ya que el USB es un Estándar de comunicación independiente de la plataforma y la Clase HID es también un Estándar. Por lo tanto, el dispositivo implementado debe funcionar de igual forma en cualquier Host que corra un sistema operativo que tenga un HID Class Driver.

Como ejemplos de uso de la Clase HID, se implementó el firmware para un teclado USB, y también el firmware para un Mouse USB. No se implementó la parte física del teclado ni del mouse porque no eran los objetivos del trabajo. A pesar de esto, se simuló el funcionamiento del teclado y el del mouse, ambos en forma satisfactoria.

4. Discusión

Debido a que la programación MCUs involucra registros de hardware (programación de bajo nivel), el único código que es migrable sin modificaciones a otro MCU es el código que corresponde a los Descriptores. Lo anterior se debe a que cada fabricante de MCUs define los registros de datos y control de la Interfaz de bus USB a su manera. Por lo anterior, es importante tener un conocimiento de cómo funciona el USB. Sobre todo en caso de tener que migrar a un MCU de otro fabricante.

En caso de necesidad de implementar un Esclavo USB que sea de otra Clase USB, habrá que verificar la disponibilidad del Class Driver en el sistema operativo del Host. Además, habrá que verificar los tipos, la cantidad, y la velocidad de los Endpoints que se necesitan para implementar dicha Clase; y, en base a estos requerimientos, seleccionar el MCU a utilizar para la implementación de la Clase.

Pueden realizarse aplicaciones que hagan uso del bus USB, pero que no estén englobadas dentro de ninguna Clase USB. Existen varias formas de hacer esto, y que no presentan mayor dificultad a la hora de escribir el firmware para el esclavo. Pero, en estos casos, deberá escribirse un Driver dedicado que haga uso de las funciones del núcleo (kernel) del sistema operativo para el control del bus USB.

Es importante hacer notar que los Reports deben siempre ser del tamaño y estructura declarados en el Report Descriptor; de otra manera, el Class Driver del sistema operativo del Host no reconocerá y descartará los Reports, aunque el Esclavo los envíe.

5. Conclusión

Se logró implementar un firmware que cumpla con las pautas de comunicación básicas establecidas para un dispositivo USB de Clase HID.

El firmware se escribió en lenguaje C.

Como aplicaciones de la Clase HID se simuló un teclado y también un mouse.

Pero, también podrían haberse simulado dispositivos tales como keypads, joysticks, indicadores, etc.

Hasta puede realizarse adquisición de datos de baja velocidad (hasta 64 Kbits/s) utilizando el hardware con Interrupt Endpoints low-speed que se utilizó y sirviéndose del HID Class Driver del sistema operativo. Utilizando MCUs que dispongan de Interrupt Endpoints full-speed, la tasa máxima podría llegar a 1,216 Mbits/s, sirviéndose del HID Class Driver del sistema operativo del Host.

Agradecimientos
-Ing. Sergio Felissia, perteneciente al grupo TDA de la UTN F.R.S.Fco.

Referencias bibliográficas

[1] Anderson, D., (2001). USB System Architecture (USB 2.0). Addison - Wesley.
[2] Axelson, J., (2001). USB Complete, 2nd Ed. Lakeview Research.
[3] Kurose, J., Ross, K, (2002). Computer Networking, 2nd Ed. Addison – Wesley.
[4] Tanembaum, A. S., (2003). Computer Networks, 4th Ed. Pearson Education.
[5] MC68HC908JB8 Technical Data, Rev 2.3, (2005). Freescale Semiconductor.
[6] Device Class Definition for Human Interface Devices (HID), (2001). USB Implementers' Forum.
[7] HID Usage Tables, (2005). USB Implementers' Forum.
[8] Universal Serial Bus Specification, Revision 2.0, (2000). Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, Koninklijke Philips Electronics N.V.

Referencias en la WWW (http://)

[1] Freescale Semiconductor: www.freescale.com
[2] USB Implementers' Forum: www.usb.org

Datos de Contacto:

- Emanuel G. Aguirre.
Institución: UTN, Facultad Regional San Francisco.
Dirección: Perú 916, San Francisco, Cba., 2400.
e-mail: ema_a_@hotmail.com.
Teléfono: 03564-422454
- Pablo Di Giulio.
Institución: UTN, Facultad Regional San Francisco.
Dirección: Pasteur226, San Francisco, Cba., 2400.
e-mail: djdshg@hotmail.com.
Teléfono: 03564-443628