



**INSTITUTO POLITECNICO NACIONAL  
CECYT NO 3  
ESTANISLAO RAMIREZ RUIZ**



**ADRIANA A**

**REPORTE DE PRACTICA 1: "PUERTO USB"**

**MANTENIMIENTO DE EQUIPO DE CÓMPUTO**

**11-FEBRERO-2009**

## USB

USB, en informática, siglas de *Universal Serial Bus*, bus serie universal. Es una interfaz de *hardware* que permite conectar periféricos de baja velocidad, como el teclado, el ratón o *mouse*, la impresora o cámaras digitales, a los ordenadores o computadoras. Cada puerto USB es capaz de gestionar hasta 127 dispositivos, cuya conexión y desconexión se podrá realizar en caliente, es decir, sin necesidad de apagar la computadora.

Los puertos USB aparecieron en los ordenadores en 1996 y se extendieron rápidamente, hasta el punto de que Windows 98 y sus versiones posteriores los soportan plenamente. Eran los denominados USB versión 1.1, cuya velocidad de transferencia de datos llegaba hasta 12 Mbps (megabits por segundo). Su éxito fue tal que en el año 2000, como resultado de una iniciativa en la que participaron Hewlett-Packard, Intel, Lucent, Microsoft, NEC y Philips, apareció el USB 2.0, una extensión del anterior, completamente compatible con aquél, que permite transmitir datos hasta 480 Mbps. Se espera que, en un futuro no muy lejano, los puertos USB reemplacen a los convencionales puertos paralelo y serie para conectar todo tipo de periféricos.

### Características de transmisión

Pin	Nombre	Color del cable	Descripción
1	VCC	Rojo	+5v
2	D-	Blanco	Data -
3	D+	Verde	Data +
4	GND	Negro	Tierra

Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos:

- Baja velocidad (1.0): Tasa de transferencia de hasta 1'5 Mbps (192 KB/s). Utilizado en su mayor parte por dispositivos de interfaz humana (Human interface device, en inglés) como los teclados, los ratones y los joysticks.
- Velocidad completa (1.1): Tasa de transferencia de hasta 12 Mbps (1'5 MB/s). Ésta fue la más rápida antes de la especificación USB 2.0, y muchos dispositivos fabricados en la actualidad trabajan a esta velocidad. Estos dispositivos dividen el ancho de banda de la conexión USB entre ellos, basados en un algoritmo de búferes FIFO.
- Alta velocidad (2.0): Tasa de transferencia de hasta 480 Mbps (60 MB/s).
- Super velocidad (3.0): Actualmente en fase experimental y con tasa de transferencia de hasta 4.8 Gbps (600 MB/s). Esta especificación será lanzada a mediados de 2009 por Intel, de acuerdo con información recabada de Internet. La velocidad del bus será diez veces más rápida que la del USB 2.0, debido a la sustitución del enlace tradicional por uno de fibra óptica que trabaja con conectores tradicionales de cobre, para hacerlo compatible con los estándares anteriores. Se espera que los productos fabricados con esta tecnología lleguen al consumidor en 2009 o 2010.<sup>1 2</sup>

Las señales del USB se transmiten en un cable de par trenzado con impedancia de  $90 \Omega \pm 15\%$ , cuyos pares se denominan D+ y D-.<sup>3</sup> Estos, colectivamente, utilizan señalización diferencial en half dúplex para combatir los efectos del ruido electromagnético en enlaces largos. D+ y D- suelen operar en conjunto y no son conexiones simples. Los niveles de transmisión de la señal varían de 0 a 0'3 V para bajos (ceros) y de 2'8 a 3'6 V para altos (unos) en las versiones 1.0 y 1.1, y en  $\pm 400$  mV en alta velocidad (2.0). En las primeras versiones, los alambres de los cables no están conectados a masa, pero en el modo de alta velocidad se tiene una terminación de  $45 \Omega$  a tierra o un diferencial de  $90 \Omega$  para acoplar la impedancia del cable. Este puerto sólo admite la conexión de dispositivos de bajo consumo, es decir, que tengan un consumo máximo de 100 mA por cada puerto; sin embargo, en caso de que estuviese conectado un dispositivo que permite 4 puertos por cada salida USB (extensiones de máximo 4 puertos), entonces la energía del USB se asignará en unidades de 100 mA hasta un máximo de 500 mA por puerto.

Miniplug/Microplug			
Pin	Nombre	Color	Descripción
1	VCC	Rojo	+5 V
2	D-	Blanco	Data -
3	D+	Verde	Data +
4	ID	Ninguno	Permite la distinción de Micro-A y Micro-B  Tipo A: conectado a tierra  Tipo B: no conectado
5	GND	Negro	Señal tierra

Los cables de datos son un par trenzado para reducir el ruido y las interferencias.



### Compatibilidad y conectores



Tipos diferentes de conectores USB (de izquierda a derecha): Micro USB, mini USB, tipo B, hembra tipo A, tipo A.

El estándar USB especifica tolerancias para impedancia y de especificaciones mecánicas relativamente bajas para sus conectores, intentando minimizar la incompatibilidad entre los conectores fabricados por distintas compañías. Una meta a la que se ha logrado llegar. El estándar USB, a diferencia de otros estándares también define tamaños para el área alrededor del conector de un dispositivo, para evitar el bloqueo de un puerto adyacente por el dispositivo en cuestión.

Las especificaciones USB 1.0, 1.1 y 2.0 definen dos tipos de conectores para conectar dispositivos al servidor: A y B. Sin embargo, la capa mecánica ha cambiado en algunos conectores. Por ejemplo, el IBM UltraPort es un conector USB privado localizado en la parte superior del LCD de las computadoras portátiles de IBM. Utiliza un conector mecánico diferente mientras mantiene las señales y protocolos característicos del USB. Otros fabricantes de artículos pequeños han desarrollado también sus medios de conexión pequeños, y ha aparecido una gran variedad de ellos, algunos de baja calidad.

Una extensión del USB llamada "USB-On-The-Go" (sobre la marcha) permite a un puerto actuar como servidor o como dispositivo - esto se determina por qué lado del cable está conectado al aparato. Incluso después de que el cable está conectado y las unidades se están comunicando, las 2 unidades pueden "cambiar de papel" bajo el control de un programa. Esta facilidad está específicamente diseñada para dispositivos como PDA, donde el enlace USB podría conectarse a un PC como un dispositivo, y conectarse como servidor a un teclado o ratón. El "USB-On-The-Go" también ha diseñado 2 conectores pequeños, el mini-A y el mini-B, así que esto debería detener la proliferación de conectores miniaturizados de entrada.

### **COMO MANEJAR EL PUERTO USB**

Se identifican 2 caminos:

- De PC a Micro
- De Micro a PC

Requisitos necesarios para manejar el protocolo USB en cualquiera de los caminos anteriores:

Conocimiento de micros de microchip serie 18 (usb)

Usar el compilador ccs

Dominar lenguaje C con APis, o Basic por lo menos

Obtener y manejar HIDComm de microchip para Basic

Visual Studio

Hardware minimo

Leer las especificaciones tecnicas del protocolo USB

### **DE LA PC AL MICRO**

Enviar y Recibir datos hacia el micro.

Lo que queremos es un puerto USB en ambiente grafico y en c que es mejor resulta muy difícil y costosa en tiempo, partiendo de eso hay varios sub-caminos:

Primero que nada manejar ventanitas en C o C++ no es nada fácil, se necesitan varias cosas un poco mas complicadas que otras, por ejemplo las que se llaman MFC Librerías, que son unas clases que lo que hacen es manejar las DLL's y las API's en una forma mas superficial pero no tan superficial como Basic.

Otro camino es una DLL creada por alguien que se aprovecha de usarla y listo. No es tan fácil porque no sabes que ha incluido ese otro ahí porque las instrucciones de DLL y librerías no son instrucciones de C comunes sino traducciones. Las DLL son una especie de enlaces a las api's de windows.

Y el otro camino mas completo y mas seguro y mas y se podría decir que es el mejor son las API's en su neta expresión. Ese es camino requiere muchos conocimientos sobre programación avanzada, lo cual se resume en tiempo, una alternativa mas fácil igual de eficiente MFC en la cual se controlan API's con mas facilidad, estas fueron desarrollados por Microsoft.

En lo que respecta al puerto usb tenemos estas opciones:

Usar DLL' S para manejar el puerto serial en ambiente grafico, lo que implica que cuando escojamos una buena DLL hay que aprender a usarla y quedarnos estancados usando solo esa, esa tiene que ser una DLL diseñada por alguien mas, o por nosotros mismos que lo que hace es manejar ventanas en c++ y le añaden control de puerto y solo el usb porque sino es muy pesada y no sirve la ejecución del programa, ya que se consume toda la memoria de ejecución.

Librerías DLL que por si solas manejen eso, por el momento hasta donde hemos visto, no hay, solo hay una que maneje el serial y otra que maneje ambiente grafico por separado y hay que unirla lo cual implica aprender a usar manejo avanzado de API'S. (Hay unas poco eficientes en Internet, pero no tienen garantía de funcionamiento, hemos probado varias).

Con el código que esta en el libro (Curso Avanzado De Microcontroladores Pic), lo que hacen es jugar con el BIOS, es una librería que lo que hace es traducir las instrucciones mas populares de ensamblador a c, pero resulta que solo funciona en Linux. Se Intento generarla y lo logramos, pero sucede que esa librería funcionan con procesos de recursividad lo que significa que dependen de otras y al tener generada la librería BIOS.H pide una \_DESF.H al tener esa pide otra y así sucesivamente y la verdad no sabemos cuando vaya a acabar, solo generamos tres pensando que hasta ahí llegaba.

Usar MFC para lograr el objetivo no es viable pues no se tiene tanto control para dominar el buffer y la única solución que hay es la de mandar un mensajito cada vez que quiera transmitir y apretar botones, etc. además de proceso engorrosos para el usuario y para el programador.

El camino más viable, eficiente, cerca y de mayor control para hacer modificaciones u aplicaciones mayores pero ya, después de sacar análisis y consultar mis fuentes fidedignas es:

Dominar las API de Windows, ahí esta la raíz de todo y a eso vamos. Entonces la conclusión es obvia todo nos lleva a las api.

Ese es el camino más eficaz y debido, sin embargo como no todo es tan bonito tenemos algunas trabas y si el factor tiempo nos agobía entonces a corto plazo presentamos esta solución:

Usar el HIDComm de Microchip que solo funciona en Visual Basic solamente para controlar el puerto, viene con 3 programas de ejemplos y hay que aprenderlo a manejar. Luego de manejarlo podemos terminar de hacer nuestra aplicación en Basic o unirla mediante un DLL con c++.

De manera que Basic solo serviría para enviar el dato al puerto usb y toda aplicación adicional que se quiera realizar descansaría sobre c++ que lo hace mas eficiente, y al unirlo con el programa en Basic mediante una dll nos ahorramos recursos de memoria porque se ejecutaría un solo proceso y no dos procesos si solo llamáramos al ejecutable de Basic.

Las opciones para Manejar el USB son CDC y HID, como se dijo anteriormente utilizaremos HID por cuestiones de conveniencia que es el protocolo de transmisión de los gamepad, Joystick, teclado, Mouse, etc.

#### Descripción del sistema USB

El USB es un bus punto a punto: dado que el lugar de partida es el *host* (PC o *hub*), el destino es un periférico u otro *hub*. No hay más que un único *host* (PC) en una arquitectura USB.

Los PC estándar tienen dos tomas USB, lo que implica que, para permitir más de dos periféricos simultáneamente, es necesario un *hub*. Algunos periféricos incluyen un *hub* integrado, por ejemplo, el teclado USB, al que se le puede conectar un Mouse USB.

Los periféricos comparten la banda de paso del USB. El protocolo se basa en el llamado *paso de testigo (token)*. El ordenador proporciona el testigo al periférico seleccionado y seguidamente, éste le devuelve el testigo en su respuesta.

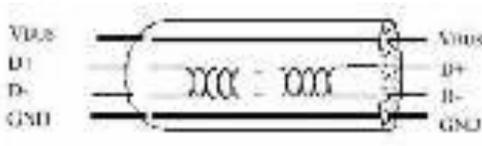
Este bus permite la conexión y la des-conexión en cualquier momento sin necesidad de apagar el equipo.

### Interfaz física

Aspecto eléctrico

Nivel eléctrico

El cable USB transfiere la señal y la alimentación sobre 4 hilos.



### Nivel de alimentación

El cable proporciona la tensión nominal de 5 V. Es necesario definir correctamente el diámetro del hilo con el fin de que no se produzca una caída de tensión demasiado importante en el cable. Una resistencia de terminación instalada en la línea de datos permite detectar el puerto y conocer su configuración (1,5 o 12 Mbits/s).

### Nivel de señal

Se trata de un par trenzado con una impedancia característica de  $90 \Omega$ . La velocidad puede ser tanto de 12 Mbits/s como de 1,5 Mbits/s. La sensibilidad del receptor puede ser de, al menos, 200mV y debe poder admitir un buen factor de rechazo de tensión en modo común. El reloj se transmite en el flow de datos, la codificación es de tipo NRZI, existiendo un dispositivo que genera un bit de relleno (bit stuffing) que garantiza que la frecuencia de reloj permanezca constante. Cada paquete va precedido por un campo de sincronismo.

### Consumo

Cada sección puede proporcionar una determinada potencia máxima siendo el PC el encargado de suministrar la energía. Además, el periférico puede estar autoalimentado (*self powered*).

### Control de consumo

El ordenador gestiona el consumo, teniendo capacidad de poner en reposo (*suspend*) o en marcha a un periférico USB. En reposo, este reduce su consumo (si puede), quedándose la parte USB funcional. Esta gestión está orientada especialmente a los equipos portátiles.

### Terminología USB

Host: Dispositivo maestro que inicia la comunicación (Generalmente la computadora).

Hub: Dispositivo que contiene uno o mas conectores o conexiones internas hacia otros dispositivos usb, el cual habilita la comunicación entre el host y con diversos dispositivos. Cada conector representa un puerto USB.

**Dispositivo compuesto:** Es aquel dispositivo con múltiples interfaces independientes. Cada una tiene una dirección sobre el bus pero cada interfase puede tener un diferente driver device en el host.

**Puerto USB:** Cada host soporta solo un bus, cada conector en el bus representa un puerto USB por lo tanto sobre el bus puede haber un varios conectores, pero solo existe una ruta y solo un dispositivo puede transmitir información a un tiempo.

**Driver:** es un programa que habilita aplicaciones para poderse comunicar con el dispositivo. Cada dispositivo sobre el bus debe tener un driver, algunos periféricos utilizan los drivers que trae Windows.

**Puntos terminales (Endpoints):** Es una localidad específica dentro del dispositivo. El Endpoint es un buffer que almacena múltiples bytes, típicamente es un bloque de la memoria de datos o un registro dentro del microcontrolador. Todos los dispositivos deben soportar el punto Terminal 0. Este punto Terminal es el que recibe todo el control y las peticiones de estado durante la enumeración cuando el dispositivo está sobre el bus.

**Tuberías (Pipes):** Es un enlace virtual entre el host (la PC) y el dispositivo USB, este enlace configura los parámetros asociados con el ancho de banda que tipo de transferencia se va a utilizar (Control, Bulk, Isochrónica o Interrupt) dirección del flujo de datos y el máximo y/o mínimo tamaño de los paquetes/buffers.

Cada enlace está caracterizado por su banda de paso (Token), su tipo de servicio, el número de punto Terminal (End Point) y el tamaño de los paquetes.

Estos enlaces se definen y crean durante la inicialización del USB. Siempre existe un enlace virtual 0 que permite tener acceso a la información de configuración del periférico USB (estado, control e información).

La norma USB define 2 tipos de enlaces virtuales (pipe); *stream* y *message*.

**Stream Pipes:** se trata de un flujo sin formato USB definido, esto significa que se puede enviar cualquier tipo de dato. Este tipo de pipe soporta las transferencias bulk, isocronas, y interrupt. Además tanto el host como el dispositivo USB pueden controlar.

**Message Pipes:** este tipo de enlace virtual si tiene un formato USB definido y solo puede soportar la transferencia Control.

### **Tipos de transferencia**

El enlace virtual (pipe) puede ser de cuatro tipos:

Control. Modo utilizado para realizar configuraciones.

Existe siempre sobre el Punto terminal 0 (EndPoint 0). Todos los dispositivos USB deben soportar este tipo de transferencia.

Los datos de control sirven para configurar el periférico en el momento de conectarse al USB. Algunos drivers específicos pueden utilizar este enlace para transmitir su propia información de control.

Este enlace no tiene pérdida de datos, puesto que los dispositivos de detección de recuperación de errores están activos a nivel USB.

### **Bulk:**

Este modo se utiliza para la transmisión de importantes cantidades de información. Como el tipo control, este enlace no tiene pérdida de datos. Este tipo de transferencia es útil cuando la razón

de transferencia no es crítica como por ejemplo, el envío de un archivo a imprimir o la recepción de datos desde un escáner.

En estas aplicaciones, la transferencia es rápida, pero puede esperar si fuera necesario. Solo los dispositivos de media y alta velocidad utilizan este tipo de transferencia.

#### Interrupt:

Modo utilizado para transmisiones de pequeños paquetes, rápidos, orientados a percepciones humanas (ratón, punteros).

Este tipo de transferencia es para dispositivos que deben recibir atención periódicamente y lo utilizan los dispositivos de baja velocidad.

Este tipo de transmisión garantiza la transferencia de pequeñas cantidades de datos. El tiempo de respuesta no puede ser inferior al valor especificado por la interfaz. El ratón o cualquier otro dispositivo apuntador es una aplicación típica de este modo de transmisión.

#### Isochronous o Flujo en tiempo real:

Modo utilizado para la transmisión de audio o video comprimido. Este tipo de transmisión funciona en tiempo real. Este es el modo de mayor prioridad.

La transmisión de la voz es un ejemplo de esta aplicación. Si ésta no se transmite correctamente, pueden llegar a oírse parásitos (*glitch*) y la aplicación puede detectar ciertos errores de los llamados *underruns*.

#### Enumeración

Cuando se conecta un dispositivo USB a la PC se produce el Proceso de Enumeración, el cual consiste en que el host le pregunta al dispositivo que se presente y le diga cuáles son sus parámetros, tales como:

- Consumo de energía expresada en unidades de Carga
- Número y tipos de Puntos terminales
- Clase del producto.
- Tipo de transferencia
- Razón de escrutinio, etc.

El proceso de enumeración es inicializado por el host cuando detecta que un nuevo dispositivo que ha sido adjuntado al Bus. El host le asigna una dirección al dispositivo adjuntado al bus y habilita su configuración permitiendo la transferencia de datos sobre el bus.