Depuración USB, problemas habituales

Total Phase, Inc.

Artículo cedido por Next For S.A., distribuidor de Total Phase, Inc. en España

www.nextfor.com

Acerca de Total Phase

Total Phase es un fabricante líder de herramientas de desarrollo para sistemas embebidos que son usados por ingenieros de todo el mundo. Total Phase se dedica a la creación de soluciones potentes, de alta calidad y asequibles para los ingenieros de sistemas embebidos. Durante años, Total Phase ha desarrollado productos que se han convertido en las herramientas elegidas por compañías de todos los tamaños, compañías del Fortune 500 y pequeñas empresas por igual. Sus clientes se encuentran en un amplio abanico de industrias como son la automovilística, la de la electrónica de consumo, los dispositivos médicos, los semiconductores, etc.

La amplia integración del USB en las aplicaciones embebidas presenta a muchos desarrolladores con el reto de trabajar por primera vez con este protocolo. La complejidad del USB puede enmascarar problemas, llevando a situaciones en las que el problema, cuando un dispositivo USB se comporta de forma incorrecta, es difícil de detectar y aislar. Los analizadores de protocolos USB permiten a los ingenieros ganar una visibilidad más profunda en el bus para ayudarles a localizar rápidamente la naturaleza concreta de un bug, ya se encuentre en la capa física, eléctrica o de protocolo. En este artículo exploraremos cómo un analizador de protocolos USB puede usarse para depurar problemas en potencia encontrados en un desarrollo USB.

Analizadores de protocolos USB

Frente a la amplia selección de herramientas de depuración como son los analizadores lógicos, osciloscopios y analizadores de protocolos, encontrar la herramienta ideal de depuración puede ser una tarea desalentadora. Afortunadamente, la complejidad del USB limita la búsqueda y selección de una solución ideal de depuración.

Como resultado de la complejidad del protocolo USB, las herramientas como analizadores lógicos u osciloscopios quedan limitadas por su visión de bajo nivel, haciendo difícil la revisión de los datos cuando se trata de un gran volumen de datos serie. En contraste, los analizadores de protocolos son capaces de monitorizar de forma no intrusiva el bus, ver datos en forma de paquetes, y capturar grandes volúmenes de datos de alto nivel específicos del protocolo. La configuración para capturar datos del bus USB es un proceso sencillo.

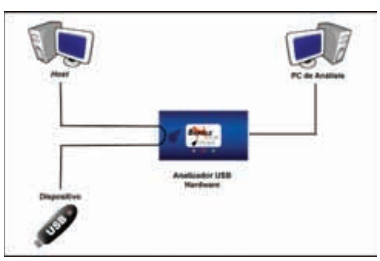

Figura 1. Configuración típica de dispositivos para la captura de datos USB

En la Figura 1, el analizador USB está conectado en línea entre el Host y el Dispositivo a monitorizar para capturar datos de forma no intrusiva. Cuando comienza la comunicación entre el Host y el dispositivo, los datos son inmediatamente enviados al ordenador de análisis que ejecuta el programa de captura para la presentación y filtrado en tiempo real de los datos.

En contraste con los osciloscopios y analizadores lógicos, el programa de captura USB puede mostrar información detallada como son las marcas de tiempo, las direcciones del device y endpoint, identificadores de paquetes (PIDs), y datos en un formato que resulta legible por el usuario. El programa también incluye funciones de búsqueda y/o filtrado para ayudar a los desarrolladores a localizar rápidamente datos de interés dentro de una gran cantidad de información.

Vamos a ver ahora algunos ejemplos de cómo un analizador de protocolos USB, como es el analizador de protocolos Beagle USB 480 (Figura 1), puede usarse para ayudar a identificar problemas habituales en un desarrollo USB.

Validez de los datos en USB

El USB utiliza dos métodos de comprobación de errores para asegurarse que los datos se envían correctamente. Junto con todas las transmisiones de datos se envía un código de redundancia cíclica (CRC) para validar la integridad de los datos dentro de un paquete. Además un toggle bit (bit alternable) se codifica en el identificador del paquete (PID) del paquete de datos para asegurarse que los paquetes se envían en la secuencia correcta. La ordenación correcta de los datos es especialmente importante cuando se intentan transferir grandes ficheros a través de múltiples transmisiones USB independientes unas de otras.

Normalmente cuando se transfieren datos en múltiples paquetes, el PID de datos alternará entre DATA0 y DATA1 en cada transmisión consecutiva que tenga éxito. En concreto, a medida que los datos son transmitidos correctamente (el CRC es válido), el receptor acusa recibo (ACK) de los datos y ambos, transmisor y receptor, alternan su bit DATA. Sin embargo, si hay un error de datos y la verificación del CRC falla, el receptor no responderá con un ACK, y el transmisor está obligado a reenviar los datos con el mismo toggle bit. El transmisor seguirá reenviando el mismo paquete de datos con el mismo toggle bit hasta que el receptor confirme la recepción correcta del paquete enviando un ACK.

En algunas ocasiones, los datos se envían correctamente pero el envío del ACK se corrompe en el bus. Cuando esto ocurre, el receptor piensa que los datos se enviaron correctamente y actualiza el toggle bit, pero el transmisor desconoce si los datos se recibieron correctamente. Por lo que el transmisor reenviará los mismos datos con el mismo toggle bit. Al no haber cambiado el toggle bit el receptor supone que se trata de una retransmisión de los mismo datos y silenciosamente ignora los datos. El receptor enviará entonces de nuevo el ACK, haciendo que el transmisor actualice correctamente el toggle bit.

Problemas en el toggle bit

Un problema habitual en USB es la gestión incorrecta del toggle bit y es un problema que resulta difícil de identificar, principalmente porque los síntomas no tienen por qué ser un dispositivo inutilizable. Un dispositivo puede funcionar con una tasa de transferencia reducida o puede que una transmisión de datos individual se pierda. Sin la ayuda de un analizador de protocolos hardware, es casi imposible deducir que la causa del problema es la incorrecta gestión del toggle bit.

Para ilustrar este problema vamos a mirar una situación en la que una aplicación en el extremo del Host no recibe ningún dato del dispositivo. Para ayudar en el proceso de depuración, el dispositivo se ha configurado para que envíe el valor de un contador que se actualiza cada vez que una transmisión tiene éxito. La raíz del problema puede deberse a una variedad de bugs relacionados con el firmware, el software y/o el hardware.

El uso de un analizador de protocolos puede localizar rápidamente este tipo de error. En las figuras 2 y 3 se están capturando datos de dos dispositivos; uno está funcionando correctamente y el otro no. En ambos casos está claro que el hardware de bajo nivel está funcionando correctamente, ya que se están transmitiendo datos válidos sin errores de CRC. Para ayudar en el análisis, se han configurado filtros software de presentación para ver únicamente los paquetes de tipo DATA en las Figuras 2 y 3.

Tras inspeccionar los registros es obvio que uno de los dispositivos no alternó el data PID (Figura 2) mientras que el otro sí alternaba el data PID tras cada paquete (Figura 3). Como mencionábamos antes, no deberían enviarse paquetes consecutivos y secuenciales con DATA0 a la aplicación, ya que el receptor ignorará los paquetes que se repiten con el mismo toggle bit. Esto explica por qué no están llegando datos a la aplicación. Sin embargo, la razón por la que se está utilizando el mismo toggle bit todavía se desconoce.

Para investigar más a fondo el asunto, podemos examinar toda la secuencia de transacciones (Figura 4). En esta vista, se ve claramente que la transacción se está realizando con éxito porque la captura muestra el ACK para cada paquete de datos, pero el bit DATA no se está alternando. Es más, las transmisiones que envían el mismo toggle bit sucesivamente se supone que envían los mismos datos. En este caso, el dispositivo no está realmente enviando los mismos datos, si no que continúa actualizando el contador. El error por consiguiente debe estar en la gestión del alternado del DATA dentro del dispositivo. Específicamente, el firmware no está alternando el bit en cada transmisión que realiza con éxito. Sin un analizador de protocolos hardware este pequeño error podría haberle costado encontrarlo al desarrollador días o semanas.

Sp	m:s.ms.us	Len	Err	Dev	Ep	Record	Data
HS	0:00.100.011	7 B		01	02	DATA0 packet	C3 00 00 00 00 FF D8
HS	0:00.100.023	7 B		01	02	DATA0 packet	C3 01 00 00 00 FE 27
HS	0:00.100.035	7 B		01	02	DATA0 packet	C3 02 00 00 00 FE 63
HS	0:00.100.046	7 B		01	02	DATA0 packet	C3 03 00 00 00 FF 9F
HS	0:00.100.058	7 B		01	02	DATA0 packet	C3 04 00 00 00 FE EB
HS	0:00.100.070	7 B		01	02	DATA0 packet	C3 05 00 00 00 FF 17

Figura 2. Los consecutivos PIDs DATA0 en la columna Record muestran que el dispositivo no está alternando el data PID.

Sp	m:s.ms.us	Len	Err	Dev	Ep	Record	Data
HS	0:00.100.011	7 B		01	02	DATA0 packet	C3 00 00 00 00 FF D8
HS	0:00.100.023	7 B		01	02	DATA1 packet	4B 01 00 00 00 FE 27
HS	0:00.100.035	7 B		01	02	DATA0 packet	C3 02 00 00 00 FE 63
HS	0:00.100.046	7 B		01	02	DATA1 packet	4B 03 00 00 00 FF 9F
HS	0:00.100.058	7 B		01	02	DATA0 packet	C3 04 00 00 00 FE EB
HS	0:00.100.070	7 B		01	02	DATA1 packet	4B 05 00 00 00 FF 17

Figura 3. El alternado de PIDs, DATA0 y DATA1, muestra el alternado correcto en los paquetes posteriores

Sp	m:s.ms.us	Len	Err	Dev	Ep	Record	Data
HS	0:00.100.010	4 B		01	02	IN txn	00 00 00 00
HS	0:00.100.010	3 B		01	02	IN packet	69 01 C1
HS	0:00.100.011	7 B		01	02	DATA0 packet	C3 00 00 00 00 FF D8
HS	0:00.100.013	1 B		01	02	ACK packet	D2
HS	0:00.100.021	4 B		01	02	IN txn	01 00 00 00
HS	0:00.100.021	3 B		01	02	IN packet	69 01 C1
HS	0:00.100.023	7 B		01	02	DATA0 packet	C3 01 00 00 00 FE 27
HS	0:00.100.025	1 B		01	02	ACK packet	D2
HS	0:00.100.033	4 B		01	02	IN txn	02 00 00 00
HS	0:00.100.033	3 B		01	02	IN packet	69 01 C1
HS	0:00.100.035	7 B		01	02	DATA0 packet	C3 02 00 00 00 FE 63
HS	0:00.100.036	1 B		01	02	ACK packet	D2

Figura 4. Una vista ampliada de la transacción muestra nuevos datos enviados con el mismo DATA toggle bit.

Sp	m:s.ms.us	Len	Err	Dev	Ep	Record	Data
FS	0:00.107.526	50.0 us				<K chirp>	
FS	0:00.107.576	50.0 us				<J chirp>	
FS	0:00.108.026	30.0 us	U			Bus event	<Reset> / <K chirp> / <Tiny K>
FS	0:00.108.056	50.0 us				<J chirp>	
FS	0:00.108.106	5.05 ms				[50 K-J Chirp Pairs] [1 K Chirp]	
FS	0:00.108.106	50.0 us				<K chirp>	
FS	0:00.108.156	50.0 us				<J chirp>	
FS	0:00.108.156	50.0 us				<J chirp>	

Figura 5. La línea resaltada muestra un evento de bus no esperado a mitad de la secuencia chirp.

Eventos de bajo nivel

Otro error frecuente ocurre con los eventos de bajo nivel del bus. La especificación USB define un número de eventos de señalización cruciales a nivel de bus, los cuales siguen una temporización y criterios específicos, y determinan cosas tan importantes como el suspend, resume y el handsake de alta velocidad. Como ejemplo, un error de este tipo podría resultar en el fallo del host en iniciar el proceso de enumeración, por lo que finalmente no podría reconocer el dispositivo USB. Este tipo de eventos son difíciles, si no imposibles, de depurar sin la ayuda de una herramienta hardware.

Un paso del handshake de alta velocidad, llamado la secuencia chirp, requiere que el host emita al menos 3 ciclos de alternar entre D- (Chirp K) y D+ (Chirp J) durante 40-60μs. Aunque las especificaciones USB sólo requieran 3 ciclos, a menudo los hosts envían cientos de estos ciclos. Mientras que es posible usar un osciloscopio para medir esta secuencia de eventos y calcular las duraciones con el cursor, dedicar tiempo de ingeniería para verificar cada chirp con un osciloscopio resultaría ser un proceso tedioso y propenso a errores. Un analizador de proto-

colos hardware, por el contrario, puede ahorrar tiempo midiendo automáticamente cada señal e indicando errores potenciales. Es más, con un analizador, estas medidas pueden realizarse en cada prueba que realicemos, capturando así de forma automática bugs nuevos o intermitentes.

En la Figura 5, el programa de captura resalta un error en un chirp con tiempos incorrectos que se ha enviado desde un host USB que está bajo desarrollo. En este ejemplo concreto, el error ocurrió a mitad de la secuencia chirp, y la revisión con el cursor en un osciloscopio no habría permitido su detección. Las consecuencias de tal error son indeterminadas, ya que el comportamiento en estos casos no está contemplado en las especificaciones, y cualquier número de procesos podrían funcionar incorrectamente. Mientras que es posible que un dispositivo sea lo bastante robusto como para seguir funcionando correctamente en esta situación, otros dispositivos pueden ser más sensibles a esta señalización fuera de las especificaciones. Si las pruebas internas sólo se realizaron con un reducido subconjunto de dispositivos y todos ellos pasaron las pruebas, es posible que el desarrollador no detecte la existencia del error si no dispone de un analizador de protocolos hardware. La salida al mercado de un producto con fallos puede resultar en la frustración de muchos clientes. Con un analizador basado en hardware los ingenieros puede evitar costes escalados por errores no detectados.

Las fase de desarrollo y depuración son pasos cruciales en el ciclo de vida de un producto. Como muestra la situación anterior, el uso de un analizador de protocolos USB basado en hardware presenta los paquetes de datos de un protocolo complicado como es el USB en un formato accesible y legible por el usuario. Por medio de un analizador los ingenieros pueden probar fácilmente sus aplicaciones e identificar rápidamente áreas problemáticas mientras reducen el tiempo de desarrollo y simplifican el proceso de depuración.