RIELAC, Vol.XXXIV 2/2013 p.1-8 Mayo - Agosto ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

Comunicación USB entre aplicaciones desarrolladas en LabVIEW y microcontroladores de Silicon Labs.

Ing. Julio César Herrera Benítez

Centro de Investigación y Desarrollo Técnico; Investigador agregado; julio@inf.cidt.mnt.

RESUMEN / ***ABSTRACT***

El presente artículo trata sobre la utilización de LabVIEW para establecer comunicación USB con microcontroladores de la familia 8051 de Silicon Laboratories, utilizando un driver desarrollado por dicha compañía. En el documento se incluye una descripción de este driver, así como de las funciones principales que permiten el manejo del mismo, las cuales se encuentran en una biblioteca de enlace dinámico. El artículo contiene además una metodología básica para el uso de estas funciones y una explicación detallada a través de un ejemplo, donde se ilustra como cargar y configurar las mismas con el ambiente de desarrollo LabVIEW. Finalmente se muestran dos ejemplos de la utilización del driver a partir de una biblioteca de funciones USB creada en LabVIEW para la comunicación con un microcontrolador.

Palabras Claves: LabVIEW, Driver, Comunicación USB, Microcontrolador, Silab, DLL.

This article is about USB communication establishes between LabVIEW and Silicon Laboratories microcontrollers 8051 family, using a driver developed by this company. This paper includes a driver description and description of the main functions that allow the management of it, which are in a dynamic link library. The article also contains a basic methodology for the use of these functions and a detailed explanation through an example of how to load and configure them using the LabVIEW development environment. Finally shows two examples of the driver use from a USB library created in LabVIEW to communicate with a microcontroller.

KeyWords: LabVIEW, Driver, USB communication, Microcontroller, Silab, DLL.

INTRODUCCION

La transmisión de datos utilizando Bus Serie Universal (USB) es de gran importancia debido a la facilidad de uso, velocidad de transmisión y su mayor presencia en las PC actuales. Silicon Laboratories es una compañía que incluye entre sus productos de base amplia a los microcontroladores 8051 con interfaz de comunicación USB.

Silicon Laboratories suministra un Kit de desarrollo USBXpress que constituye una herramienta para la comunicación USB entre los dispositivos de las familias C8051F32x, C8051F34x, C8051F38x, CP210x y la PC. Esta herramienta incluye la creación de un driver para Windows, instalador del driver que se crea, librería de funciones API (Application Program Interface, por sus siglas en inglés) para el host que se encuentran en forma de DLL (Dinamic Link Library, por sus siglas en inglés), y funciones para la interfaz del firmware (estas solamente para dispositivos C8051F32x/34x/38x) [6].

LabVIEW constituye un revolucionario entorno de programación gráfica para aplicaciones que involucren adquisición, control, análisis y presentación de datos. Una de las principales ventajas que proporciona el empleo de LabVIEW es la posibilidad de

incorporar aplicaciones escritas en otros lenguajes y aplicaciones tales como: DLL (librerías de funciones), .NET, ActiveX, Multisim, Matlab/Simulink, entre otras [1] [2].

El objetivo de este trabajo es mostrar cómo establecer una comunicación con los microcontroladores de Silab a través de bus USB para el desarrollo de aplicaciones que requieran el trasiego de datos desde y hacia la PC. Además que esta comunicación se realice con el software de aplicación LabVIEW de forma eficiente y que permita la reutilización del código por otros usuarios.

CONTENIDO

De forma abreviada, la comunicación entre la interfaz del host con el dispositivo USB (Figura 1) se realiza a través de las funciones API que se encuentran en la DLL. Estas funciones API son las que manejan el driver USBXpress, este a su vez interactúa con el firmware del dispositivo. El firmware opera el controlador USB de los microcontroladores a *"Full Speed"* (12 Mbps) y usa transferencias tipo Bulk.

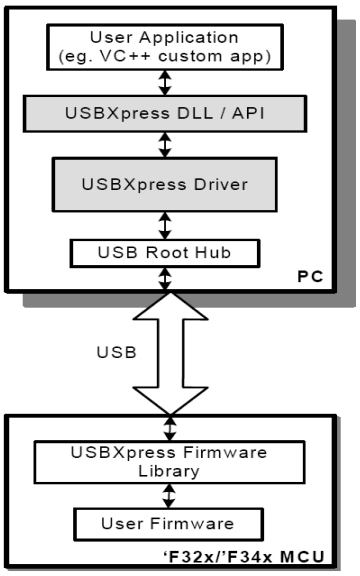

Figura 1. Flujo de datos entre el host y el dispositivo.

FUNCIONES DE LA API

La biblioteca dinámica que proporciona *Silab* contiene una gran cantidad de funciones, entre las más utilizadas podemos encontrar:

- Función para saber el número de dispositivos USB conectados (que usan ese driver). SI_GetNumDevices().
- Función que devuelve los descriptores USB del dispositivo. SI_GetProductString().
- Funciones que permiten abrir y cerrar una instancia con el dispositivo. SI_Open(), SI_Close().
- Funciones que permiten la lectura y escritura desde o hacia el dispositivo USB. SI_Read(), SI_Write().
- Funciones que devuelven el estado y el número de bytes en la cola de recepción. SI_CheckRXQueue().

De forma general para utilizar estas funciones e iniciar una comunicación con un dispositivo USB, el usuario primero debe utilizar la función *SI_GetNumDevices()* para saber si existe algún dispositivo conectado. Si hubiese uno conectado, entonces esta función nos devuelve un valor distinto de cero por lo que el usuario puede, con la función *SI_GetProductString()*, conocer los parámetros del descriptor tales como: número de serie, descripción del producto, ID del producto y del fabricante, entre otras [5]. Estos

descriptores del dispositivo USB son personalizados mediante el conjunto de funciones de interface durante la programación de microcontrolador.

Como se detectó un equipo conectado entonces es posible establecer una comunicación con el mismo, para ello se utiliza la función *SI_Open()* que le devuelve al usuario un *Handle*, utilizado en las operaciones subsiguientes [5].

La transmisión de los datos se hace utilizando las funciones *SI_Read()* y *SI_Write()* para las operaciones de lectura y escritura respectivamente desde y hacia el microcontrolador. Para saber cuándo efectuar la lectura el usuario puede auxiliarse además, de la función *SI_CheckRXQueue()* que retorna el estado de la cola de recepción y el número de bytes presentes en la misma. Puede auxiliarse también de la función *SI_FlushBuffers()* que limpia los buffers de transmisión en el dispositivo y de recepción en la PC. En caso de no realizar ninguna otra acción se cierra la comunicación con la ayuda de la función *SI_Close()* [5].

COMO UTILIZAR LAS FUNCIONES DE LA API CON LABVIEW

Como se dijo anteriormente, el LabVIEW proporciona algunas opciones para la interconexión con códigos creados en otros lenguajes. Una de las formas más comunes de encontrar código externo es en forma de DLL, las cuales pueden ser ejecutadas por el LabVIEW haciendo uso de *Call Library Function Node* (Figura 2). Con esta función nodo se puede hacer uso de la biblioteca proporcionada por *Silab*, definiéndosele la función API a utilizar.

Figura 2. Función de LabVIEW para utilizar DLLs.

Para configurar la función anterior se da doble clic sobre la misma e inmediatamente se despliega una ventana que permite llenar los campos que la componen (Figura 3). En la pestaña *Function* se introduce la dirección de la DLL de *Silab* así como se selecciona la función a utilizar, en este caso vamos a seleccionar como ejemplo la función de escritura *SI_Write*.

También se especifica si la función puede ser llamada de forma reentrante, es decir, que pueda ser ejecutada varias veces de forma simultánea. En sistemas operativos multi-hilos es posible hacer múltiples llamados a una DLL simultáneamente. Para que la función se pueda ejecutar de forma reentrante se selecciona la opción *"Thread"* como *"Run in UI Thread"*, que establece o crea un hilo de ejecución independiente para la función [1 3].

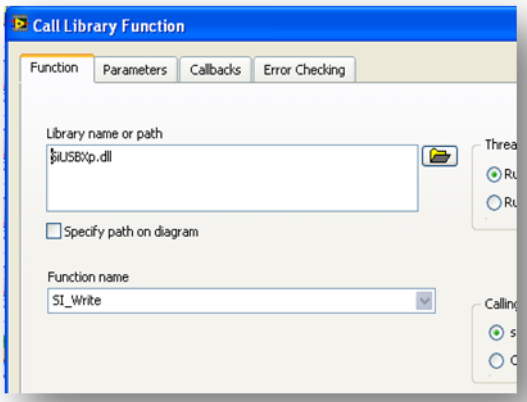

Figura 3. Selección de la función en la DLL.

En la pestaña *Parameters* de la misma ventana se van agregando los parámetros de entrada y salida a dicha función con sus tipos de datos (Figura 4). En el caso del ejemplo que se viene implementando de la función *SI_Write*, se le definen los valores con sus respectivos campos:

Valores de entrada

1. El Handle que nos devolviera la función *SI_Open*.
   a. Type: Numeric.
   b. Data Type: Unsigned 32-bit Integer.
   c. Pass: Value.
2. El buffer con los datos a escribir.
   a. Type: Array.
   b. Data Type: Unsigned 8-bit Integer.
   c. Array Format: Array Data Pointer.
3. El tamaño en bytes de los datos a escribir.
   a. Data Type: Unsigned 32-bit Integer.
   b. Pass: Value.

Como valor de salida devuelve:

1. La cantidad en bytes de datos escritos.
   a. Type: Numeric.
   b. Data Type: Unsigned 32-bit Integer.
   c. Pass: Value.
2. El valor de retorno.
   a. Type: Numeric.
   b. Data Type: Unsigned 32-bit Integer.

El valor de retorno nos indica si la operación tuvo éxito o no, este es un valor codificado que ante la ocurrencia de un problema nos informa de su procedencia.

Toda esta operación se hace con ayuda de la información suministrada para la DLL, además se debe tener en cuenta los tipos de datos de C y su correspondencia en LabVIEW. En [4] se puede encontrar más información referente a los tipos de datos que soporta el LabVIEW provenientes de librerías compartidas.

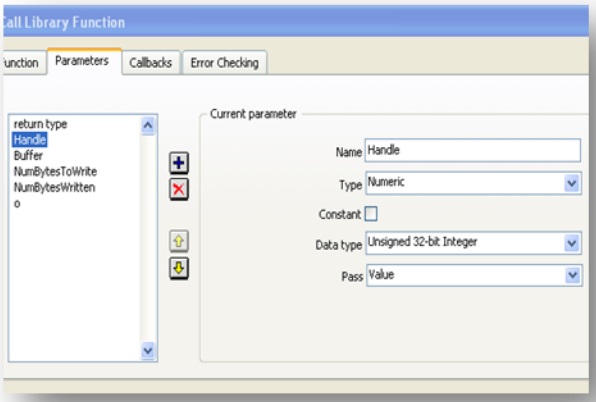

Figura 4. Selección de los campos de entrada y salida a la función.

Finalmente la función de escritura quedaría parametrizada como se muestra en la Figura 5. Solo resta guardarlo como un SubVI para su reutilización futura. Estos subVI pueden ser configurados también para ser llamados de forma reentrante, para ello solo hay que ir a las propiedades del mismo y en la categoría *"Execution"* seleccionar *"Reentrant execution"*.

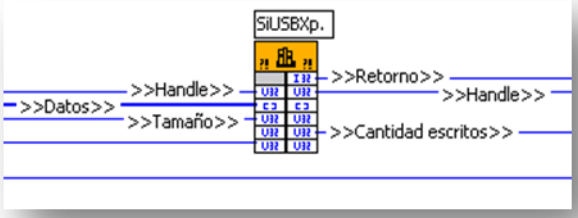

Figura 5. Función SI_Write parametrizada.

BIBLIOTECA DE FUNCIONES USB EN LABVIEW

Si se repite el proceso del epígrafe anterior para cada una de las API deseadas, se obtienen una serie de SubVIs que pueden ser utilizados en la creación de una biblioteca personalizada de funciones USB. Por ejemplo, se toman los SubVIs creados y se introducen dentro de una carpeta que puede ser llamada "USB". Está carpeta puede ser incluida como una subpaleta dentro de la paleta principal de funciones de LabVIEW, copiando la misma en la carpeta de componentes de usuario, que se encuentra en la dirección donde se instala el software de *National Instruments*. Finalmente quedaría como se muestra en la figura 6.

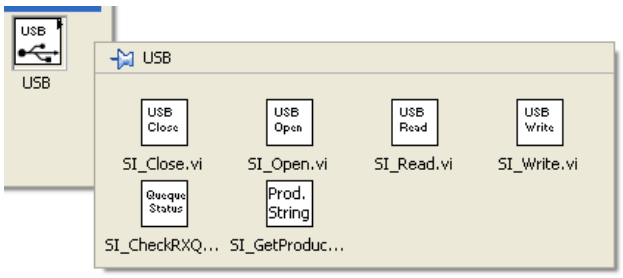

Figura 6. Subpaleta de funciones USB.

En la figura 7 se muestra un ejemplo de posible código de uno de los subVI que conforman esta biblioteca, específicamente el que realiza la operación de lectura. Una vez que se establecen los parámetros de entrada y salida de la función *"SI_Read"* en el *Call Library Function Node*, se le agregan entonces los controles e indicadores que representan las entradas y salidas de este subVI. A la función de lectura se le pasa el handle y el número de bytes a recibir, y devuelve el handle, el valor de retorno, el buffer con los datos leídos en bytes y la cantidad de bytes retornados. Este último parámetro se utiliza para verificar que la cantidad de datos en bytes fijada por el usuario coincide con la cantidad real de datos adquiridos, si no hay coincidencia entonces el indicador booleano toma valor verdadero.

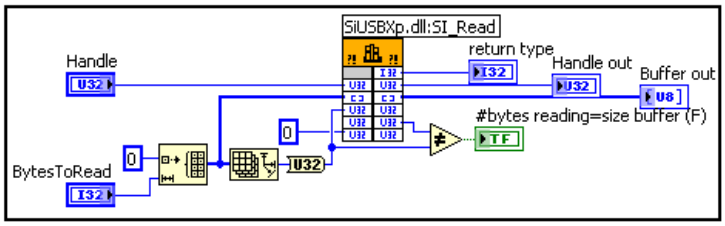

Figura 7. Código de subVI SI_Read.

EJEMPLOS

En este acápite se muestran dos ejemplos prácticos del uso que tienen los VIs creados para la comunicación con uno de los micros de *Silab* con interfaz de comunicación USB, supongamos el *C8051F320*.

El primero es un ejemplo de **lectura** de datos enviados desde el microcontrolador (Figura 8). Supongamos que el *C8051F320* se encuentra midiendo los valores de temperatura de cuatro sensores ubicados en el interior de un local. El microcontrolador digitaliza periódicamente los valores de tensión de los cuatro sensores y los envía por USB hacia la PC. Para recibir estos valores, lo primero que debemos hacer es crear una nueva instancia a partir de la existencia de un dispositivo conectado utilizando el VI *SI_Open* que nos devuelve un Handle. Siguiendo el flujo de datos de la figura 7, se entra a un ciclo de espera donde se utiliza el VI *SI_CheckRXQueue()* para determinar la cantidad de bytes presentes en la cola de recepción. Este es un VI al que se le pasa como entradas el handle obtenido y la cantidad de bytes a chequear, y nos devuelve la cantidad de bytes contenidos en la cola, un booleano que es verdadero si esta cantidad es mayor o igual a la cantidad a chequear y el valor de retorno para verificar si la operación es correcta. En el ejemplo si el booleano se hace verdadero es que están los bytes en la cola listos para leerlos y por lo tanto se sale del ciclo de espera. Se realiza la lectura con el VI *SI_Read()*, descrito en epígrafe anterior. En [5] se aclara que el número máximo de bytes a leer desde el dispositivo en un ciclo no puede ser mayor que 64 kB. Con los bytes leídos se puede realizar cualquier operación de procesamiento o simplemente indicar los valores de temperatura en controles del panel frontal de la aplicación. Terminada la lectura se cierra la comunicación con el dispositivo empleando el VI *SI_Close*.

El segundo es un ejemplo de **escritura** de datos hacia el microcontrolador (Figura 9), donde al igual que en el caso anterior usamos las funciones *SI_Open* y *SI_Close* para abrir y cerrar la comunicación con el micro. Supongamos ahora que en este ejemplo queremos enviar la información de control correspondiente a una o varias válvulas que el microcontrolador maneja. Para ello, una vez obtenido el handle y siguiendo el flujo de datos se ejecuta el VI *SI_Write* que llama la función de escritura, al mismo se le pasan como valores de entrada el handle anterior y los datos en forma de arreglo U8 (entero sin signo de 8 bits). Este VI tiene como valores de salida un booleano que permite verificar la veracidad de la escritura, es decir, que toma verdadero si la cantidad de datos a escribir fue igual a la cantidad escrita. El VI *SI_Write* también devuelve el valor de retorno de la función que utilizamos para saber si la operación fue correctamente ejecutada. En [5] se dice que el número máximo de bytes a escribir en un ciclo no puede ser mayor que 4096 Bytes.

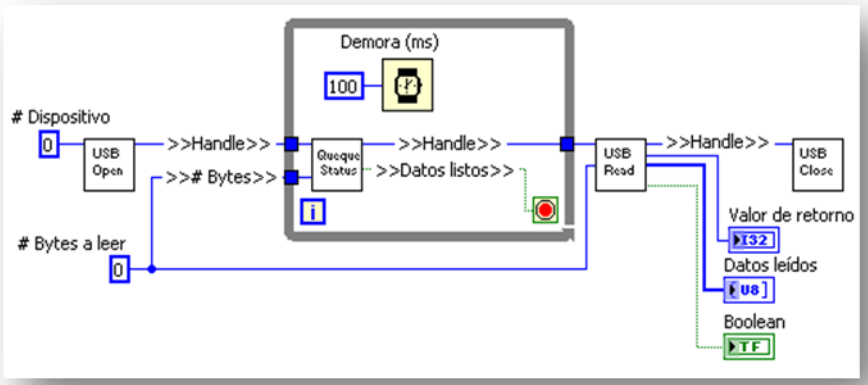

Figura 8: Ejemplo de Lectura.

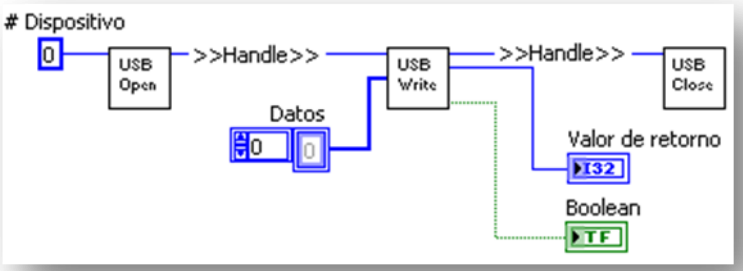

Figura 9: Ejemplo de escritura.

CONCLUSIONES

LabVIEW es un software que permite ejecutar código externo y una de las funciones que utiliza para ello es *Call Library Function Node*. Para la configuración de este, se debe seleccionar la función de la DLL a utilizar y se le agregan los parámetros de entrada y salida con sus tipos de datos.

A partir de la configuración de estas funciones es posible crear una biblioteca de subVIs en LabVIEW, que permitan establecer una comunicación con la familia de *Silicon Laboratories* que tienen bus de comunicación USB (C8051F32x y C8051F34x).

Esta Biblioteca creada en LabVIEW, que utiliza las funciones de la DLL suministrada por *Silab*, puede ser de gran ventaja ya que permite acortar los tiempos de programación y la reutilización del código por diferentes usuarios en gran cantidad de aplicaciones con microcontroladores de esta familia.

REFERENCIAS BIBLIOGRÁFICAS

1. TRAVIS, J.; KING, J. "LabVIEW for Everyone", 3rd Edition. Prentice Hall, 2007. 981 p. ISBN 0-13-185672-3

2. BITTER, R.; MOHIUDDIN, T.; NAWROCKI, M. LabVIEW™ Advanced Programming Techniques. 2nd Edition.CRC Press, 2007. 499 p. ISBN 0-8493-3325-3

3. National Instruments, "Using External Code in LabVIEW", LabVIEW ™, April 2003

4. National Instruments. LabVIEW Help "Supported Data Types for the Import Shared Library Wizard"

5. Silicon Laboratories Inc., "USBXPRESS® PROGRAMMER'S GUIDE", Application Note 169, revision 2.0. 2008.

6. Silicon Laboratories. USBXpress® Development Tools. Visitado: 16-4-2012. Disponible en: www.silabs.com /products/mcu/Pages/USBXpress.aspx

AUTOR

Julio César Herrera. Graduado en ingeniería automática del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) en el año 2009, sito en la Habana, Cuba. Centro de Investigación y Desarrollo Técnico (CIDT); Investigador agregado; julio@inf.cidt.mnt.