CIRCUIT CELLAR ONLINE

FEATURE ARTICLE

Don Powrie

Jump on the Bus

Taking a Look at the USB

Wanting to use the USB but lacking in the knowledge and experience necessary to create a device driver can cause some aggravation. When Don realized that there was a limited supply of information in this area, he had to come up with another solution. In this article, he takes us through that solution and shows us that a little knowledge goes a long way.

Several manufacturers make silicon for implementing a USB 1.1 compliant interface. Some include built-in microcontrollers, and some are simply the serial engine. However, no matter what they have or don't have, they all include detailed datasheets that are stuffed full of valuable information. Some even have application notes that describe in detail how to connect to a microcontroller or your target electronics. USB is a great idea in that the electronics are easy to implement. Almost all new PCs have the interface built in, and being able to connect and disconnect a USB device to a PC without shutting down is convenient.

However, the one topic that seems to be completely avoided is information about the inevitable device driver, without which USB devices simply will not operate. The device driver is the bridge between the application software running on the host PC, the USB port on the PC, and the USB silicon out at the end of the USB cable.

Device drivers are tricky pieces of software. Only a small percentage of programmers have the knowledge and experience necessary to create device drivers. A driver that is not written well can result in the blue screen of death or complete system lockup. Some operating systems assume all device drivers are well behaved and make no attempt to protect themselves from any errors in the driver.

One solution to this problem is the FT8U245AM and virtual COM port drivers from FTDI. After the virtual COM port (VCP) drivers are installed, the application software merely has to open a COM port and read or write as though it were talking over standard RS-232. The VCP drivers intercept the data that would otherwise go to the RS-232 port and feed it to the USB scheduler, which then sends it to the USB port. Setting the baud rate in the application program has no effect on the data rate. The FT8U245AM always communicates at the maximum data rate. After a FT8U245AM is connected to your system and the drivers are loaded, select which COM

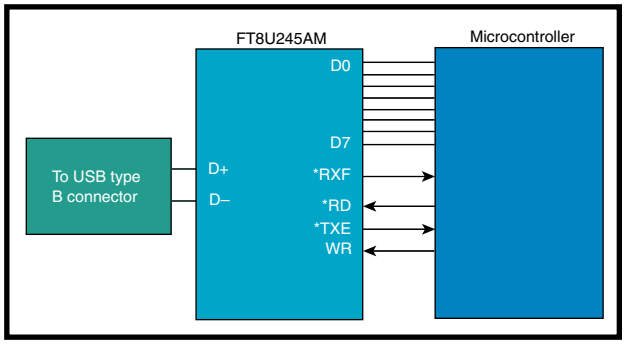

Figure 1—*The FT8U245AM communicates with a target microcontroller via an 8-bit data bus and four handshaking lines.*

port your application program will access via the System Properties page.

SPEED LIMIT

Version 1.1 of the USB specification outlines two data rates—low speed, which transfers data at 1.5 Mbps, and full speed, which transfers data at 12 Mbps. The FT8U245AM transfers data at a maximum of 8 Mbps, somewhat slower than full speed but still much faster than the RS-232. In order to achieve the 8-Mb data rate, the target microcontroller must be able to read and write data every 125 ns. This is a tall order for most microcontrollers unless they are running at high clock speeds.

Another limiting factor for most USB devices is the 1-ms frame. USB devices transfer data in packets. If data is to be sent from the PC, a packet is built up by the application program and sent via the device driver to the USB scheduler. This scheduler puts a request to the list of tasks for the USB host controller to perform. This will typically take at least 1 ms to execute because it will not pick up the new request until the next 1-ms USB frame.

Therefore, there is a sizable overhead, depending on your required throughput, associated with moving the data from the application to the USB device. If data is sent one byte at a time by an application, the overall throughput of the whole system will be severely limited. The message here is that, if speed is critical, always send data in packets.

The specification for USB 2.0 is a good example of just how complicated a communications protocol can be. [1] It's easy to get lost in the myriad of details that outline USB communications. Thankfully, FTDI's drivers hide most of these details and provide an easy to use interface.

DLL-BASED DRIVER

With some programming languages, you can communicate via RS-232 ports (including the VCPs) by using a statically-linked communications library that handles the interrupts and provides an easy programming interface (open, read, write, etc.). Using this type of communications library presents an extra step, which you, purely for simplicity's sake, might like to avoid.

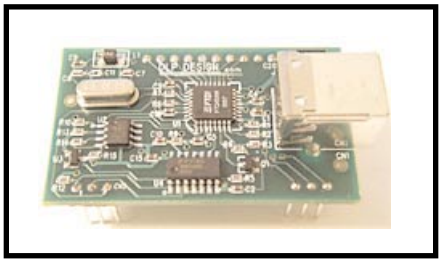

Photo 1—*The DLP-USB1 from DLP Design allows you to easily evaluate the FT8U245AM. Mounting pins on 0.1″ spacings makes easy work of interfacing to target electronics.*

FTDI recently released a new version of their drivers that is based on a dynamically-linked library (DLL). The application program simply requests the operating system to load the DLL at runtime, calls the open function to connect to the FT8U245AM device, and starts reading and writing data to and from the FT8U245AM's FIFO memory. No other statically-linked communications library is required. Programming examples for how to use the DLL with various compilers are available for download from FTDI's web site.

THE MECHANICS

The FT8U245AM is available from FTDI in surface-mount form as a 32-pin MQFP, and samples of the FT8U245AM are available through Saelig. The evaluation board (DLP-USB1) can be seen in Photo 1.

Interfacing to the FT8U245AM is made via eight data lines and four handshaking lines, as outlined in Figure 1 and Table 1. The FT8U245AM's internal FIFO is comprised of two buffers, which can hold 128 bytes of received data coming from the host PC and 384 bytes of data to be transmitted to the host.

Additional circuitry can be used to detect when the device enters Standby mode. This is usually nothing more than a quad NAND gate.

Power for your target electronics can be taken from the USB port, provided you are careful not to exceed the limits of 500 mA during normal operation and 500 µA when in Standby mode. The FT8U245AM enters Standby mode after 3 ms of no USB activity (meaning there is no start of frame packets). Meeting this power specification will require that you shutdown your target electronics when the FT8U245AM enters Standby mode, most likely by adding a MOSFET power switch or switches.

The product ID (PID), vendor ID (VID), device description, and manufacturer name can be stored in an EEPROM device via a built-in interface on the FT8U245AM. The EEPROM can also store a unique serial number that is generated and written by the program 232PROG.EXE, which

Pin	Direction	Function
*RD	Input	When pulled low, *RD takes the eight data lines from a high impedance state to the current byte in the FIFO's receive buffer. Taking *RD high returns the data pins to a high impedance state and prepares the next byte (if available) in the FIFO to be read.
WR	Input	When taken from a high state to a low state, *WR reads the eight data lines and writes the byte into the FIFO's transmit buffer. Data written to the transmit buffer is immediately sent to the host PC and placed in the RS-232 buffer, which is opened by the application program.
*TXE	Output	When high, the FIFO's 384-byte transmit buffer is full or busy storing the last byte written. Do not attempt to write data to the transmit buffer when *TXE is high.
*RXF	Output	When low, at least one byte is present in the FIFO's 128-byte receive buffer and is ready to be read with *RD. *RXF goes high when the receive buffer is empty.

Table 1—The FT8U245AM's four handshaking lines give the target microcontroller complete access to the FIFO buffer.

can be downloaded from FTDI's web site. If you are planning to commercialize a product with a USB port, you must register your own PID and VID with the USB-IF.

USB analyzers are available from a number of companies and tend to be rather expensive, but they are worth the money if you are attempting to isolate an elusive software bug. The level of information that is provided by these analyzers requires that you be extremely familiar with the USB specification. At this level of understanding, you will probably want to write your own drivers.

DESTINATION

USB devices are becoming a common topic in computer magazines and are widely available on shelves at your local computer shop. The ability to connect and disconnect to a PC without shutting down can be convenient and provide for designs like flash memory devices that reside on your key chain. Although RS-232 and parallel printer ports will probably never completely disappear from the back of most PCs, the devices that currently connect to these legacy ports most likely will in favor of an inexpensive device with a USB interface.

Don Powrie has been developing hardware and software for over 17 years. His forte is Visual C++ for Windows and embedded C for microcontrollers. He is the owner of DLP Design, a sole proprietorship specializing in the creation of tools to help engineers and hobbyists embrace USB technology. He may be reached at don@dlpdesign.com.

SOURCES

FT8U245AM and Virtual COM port drivers
Future Technology Devices Intl. Ltd. (FTDI)
+44 141 353 2565
Fax: +44 141 353 2656
www.ftdichip.com

Saelig
(716) 425-3753
Fax: (716) 425-3835
www.saelig.com

DLP-USB1 Evaluation board
DLP Design
(858) 513-2777
Fax: (858) 513-2777
www.dlpdesign.com

REFERENCE

[1] USB Implementers Forum, Inc., USB Revision 1.1 Specification, www.usb.org/developers/docs.html.