ELI FLAXER
TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING, TEL-AVIV, ISRAEL
flaxer@afeka.ac.il

Build A Device Emulator Around An Off-The-Shelf Universal Serial Bus Bridge

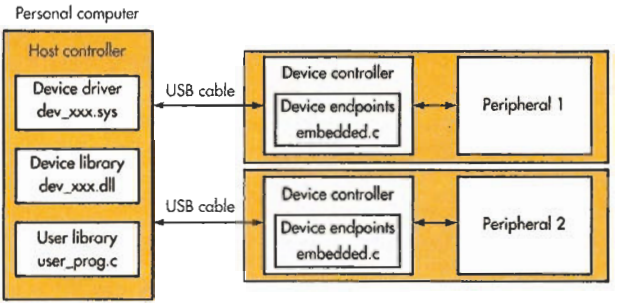
From its introduction in 1995, the Universal Serial Bus (USB) quickly gained widespread acceptance for connecting peripherals to personal computers. More recently, its ease of use, expandability, high bandwidth, and low cost have suited it quite well for data transfer in embedded consumer electronic and mobile devices. Thus, product developers are increasingly being asked to design and implement a wide range of USB I/O devices. This idea describes a technique for developing such devices.

A USB system is asymmetric, consisting of a host controller with several downstream USB ports and multiple peripheral devices connected in a tiered-star topology *(Fig. 1)*. The host (usually a personal computer) must include a device driver (xxx.sys file) to communicate with the device controller. In addition, an application programming interface (API) is needed (xxx.dll file) to connect the application to the device driver. Finally, the device controller contains an embedded program that defines the device endpoint and mode of operation. Peripheral designers must develop those three software elements.

An alternative is to use an off-the-shelf USB bridge like the FT245R from Future Technology Devices International (FTDI). The bridge includes the silicon and all software elements.

An evaluation module of this bridge (UM245R) has a DIP24W footprint and is configured for one endpoint of an 8-bit data bus (DB) and four control bits (WR, RD, TXE, and RXF). The user needs to connect the bridge to the peripheral processor or to the state machine (FPGA) that responds to the bridge protocol to implement the entire design. Then, the developer must check the communication between the peripheral and the software that runs on the host.

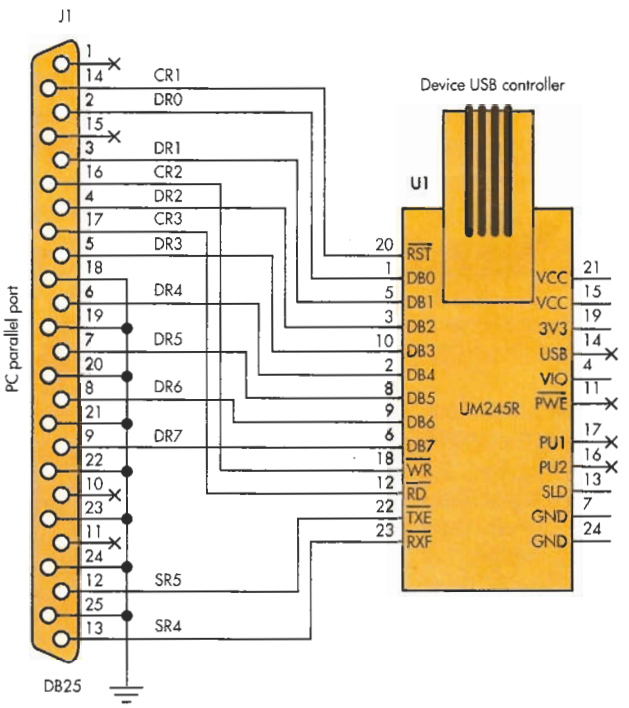
ELI FLAXER, associate professor of computer science and electrical engineering, holds a BSc in physics and computer science, an MSc in applied physics and electrical engineering, and a PhD in applied physics from Tel-Aviv University, Israel.



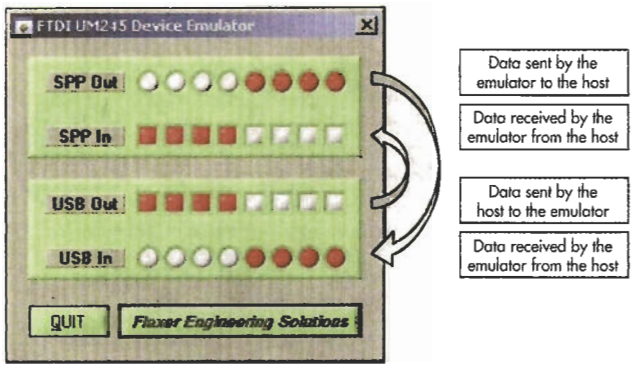
1. USB devices connect to the host via a tiered-star topology through a device controller that employs an embedded program to define the device endpoint and mode of operation.

In most cases, the developer must start debugging before the peripheral hardware is available. Because the PC parallel port in standard parallel port (SPP) mode is a real bidirectional port, you can use it to emulate the peripheral processor. Figure 2 illustrates the connections between the personal computer's parallel-port connector and the UM245R.

The parallel port's data register (DR[0...7]) connects to the module data bus (DB[0...7]). The parallel port control register (CR[1...3]) connects to the reset (RST), write (WR), and read (RD) control pins of the module, while two parallel port status bits (SR[4, 5]) connect to the receiving flag (RXF) and transmit enable (TXE) control pins of the module.



2. As a bidirectional port, the PC's parallel port can be used to emulate the peripheral processor. In this example, the UM245R evaluation module acts as an off-the-shelf USB bridge between the host and the device.



3. The emulator's graphical user interface, created using LabWindows CVI version 8.5, illustrates the unit's operation.

We used National Instruments' LabWindows CVI to implement the emulator in C. The three different source files (UM245SppLib.c, UM245UsbLib.c, and UM245Tester.c) and additional source material are available with the online version of this article at *www.electronicdesign.com*, ED Online 20575.

UM245SppLib.c includes all of the functions that access the module through the parallel port—that is, reading and writing a byte, setting and clearing the control bits, and reading the status bits. UM245UsbLib.c includes functions that encapsulate several initialization functions from FTDI's API library—ftd2xx.dll. The third file, UM245Tester.c, is the emulator main module. To work with the FTDI API library, programmers should add the ftd2xx.lib file to their project and copy the ftd2xx.dll to the working directory.

To illustrate the emulator's operation, we built a graphical user interface (GUI) that includes four buses: SPP Out, SPP In, USB Out, and USB In *(Fig. 3)*. When setting a bit in the SPP Out buttons, the emulator read the entire eight bits and sent them, as a byte, to the module through the parallel port. This action emulates data transmission by the device to the host. In response, the host software received the byte and represented it on the USB In indicators.

When setting a bit in the USB Out buttons, the host software read the entire eight bits and sent them, as a byte, to the module through the USB. In response, the emulator received the byte and represented it on the SPP In indicators. This action emulates data transmission by the host to the device.