By Don L. Powrie

ADD A UNIVERSAL SERIAL BUS INTERFACE TO YOUR NEXT PROJECT -- IT'S EASIER THAN YOU MIGHT THINK!

If you can write Windows application software that can open, read from and write to the PC's RS-232 serial ports, then you already know enough to incorporate Universal Serial Bus (USB) into your next hardware design!

Until now, interfacing a new hardware design to a PC meant connecting it to either the serial or parallel port. Serial ports offer a maximum data rate of about 230K bits per second. Parallel printer port interfaces are faster but tie up the much needed printer port and present a bit of a programming challenge. Now with the introduction of a nifty little chip and driver software developed by FTDI of Scotland (www.ftdi.co.uk), hobbyists needing a fast, easy connection to the PC with a data rate of up to 8 megabits per second can use USB.

FTDI's FT8U245AM makes designing a USB 1.1 compliant hardware interface easy. But, as some of you well know, the hardware design for most USB interfaces is trivial when compared to the effort that goes into developing the Windows driver software. This is where FTDI rises above the rest of the USB silicon manufacturers with their virtual COM Port drivers. As stated in the FT8U245AM datasheet, "By using FTDI's virtual COM Port drivers, the peripheral looks like a standard COM Port to the application software. Commands to set the baud rate are ignored--the device always transfers data at its fastest rate regardless of the application's baud rate setting."

Once a FT8U245AM is connected to your system and the drivers are loaded, you select which COM port you want to use by going to the Device Manager via the System Properties page. Next, open the "Ports" selection, right click on USB Serial Port (COMx), select properties, select the Port Settings tab, click on Advanced and there you can change the COM port number. Note, however, that you cannot change the COM port number until a FT8U245AM is connected to the system. Windows will not allocate a resource to a USB device until it is connected.

Device Drivers

A Device Driver is a highly complex piece of software that serves as a mediator between the Windows operating system, an application program and a piece of hardware. A driver defines a communications protocol that is used to access the functionality of a hardware device in a well-defined manner.

For example, many of the new webcams available at your local computer shop come with a USB interface. When you connect the webcam's cable to the PC's USB port for the first time,

Windows has no idea how to pass data between the webcam's electronics and its application program running on the host PC. Since the application program cannot "talk" directly to the PC's USB hardware, a Device Driver is required. When the webcam is plugged in, Windows will ask the unknown piece of hardware for identification. The webcam responds with its PID and VID (more on these later), and Windows will then search for the correct driver assigned to that particular camera. If Windows cannot find the appropriate driver (as it would, say, for a standard keyboard or mouse), it will request that the user provide a driver (typically found on a floppy disk that came with the webcam). Once the driver is installed, the application program can then be run, and the webcam puts your smiling face on the screen.

A Device Driver is much more complex than you might imagine. A poorly written device driver can easily crash the most stable of operating systems. No effort is made by the operating system to protect itself from a poorly written device driver. Once loaded, a device driver becomes an integrated part of the operating system's kernel.

Due to this level of complexity, only a very small percentage of programmers have the knowledge and experience necessary to write Device Drivers. Thanks to FTDI's virtual COM drivers and the FT8U245AM, all you have to know to use USB on your next project is how to open, read and write to the RS-232 ports.

Implementation

The FT8U245AM is only available from FTDI in surface-mount form. Soldering the device is a bit tricky in that it requires a steady hand and a fine tipped soldering iron. Even my best attempts at soldering the chip by hand resulted in some cleanup work with de-soldering braid. Another easy way to add this device to your project is to buy the DLP-USB1 module (Figure 1) from www.dlpdesign.com.

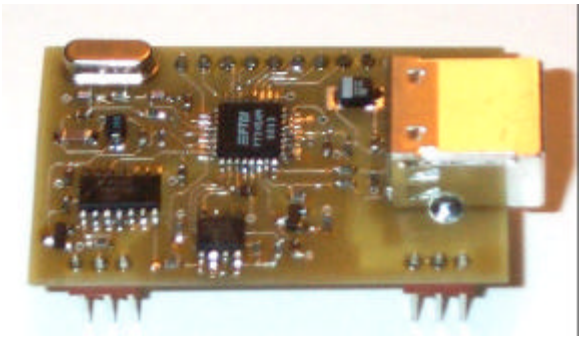
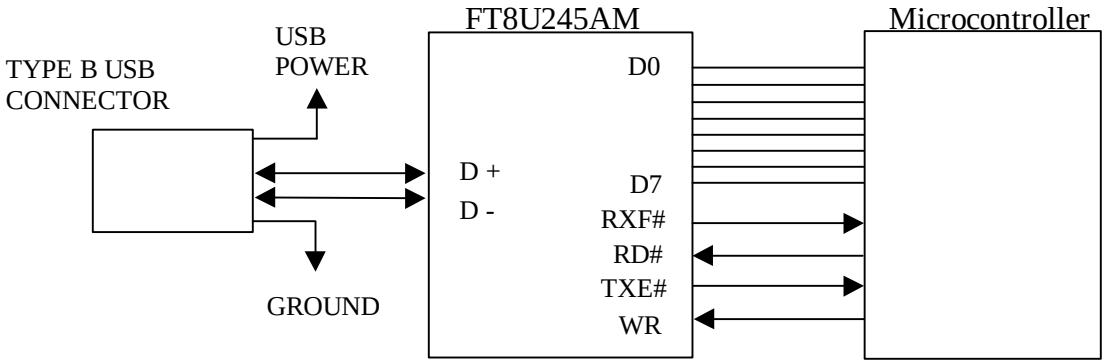


Figure 1 - DLP-USB1 from www.dlpdesign.com

Four basic hand-shaking lines and eight data lines D[7..0] are provided to interface with the chip. 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.



RD# (input) When pulled low, RD# takes the 8 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 8 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 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 1 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.

Connection to a microcontroller is made with these 12 lines. A quad NAND gate can be used to detect when the FT8U245AM is in sleep mode (suspend) and provide a reset line. Refer to DLP-USB1's Schematic (Figure 2) for the SLEEP# and RESET# signals.

Be careful not to use any power provided via the USB cable from the host unless you are fully versed in the USB specification. The maximum current available from the upstream port (or hub) when active is 500 milliamps, and this drops to only 500 microamps when in the suspend state. If the FT8U245AM does not observe bus activity for 3 or more milliseconds, then it is required to enter suspend mode. If your target electronics are getting power from the USB port, then they must be shut down when the FT8U245AM enters suspend mode. Your best bet here is to not take any power from the USB port for your target electronics. The DLP-USB1 adapter includes a connector pin that provides power for target electronics but I really must discourage the use of USB port power unless you can ensure your design meets the power requirements.

USB Implementers Forum (USB-IF)
As stated on their website (www.usb.org), USB Implementers Forum, Inc. is a non-profit corporation founded by the group of companies that developed the Universal Serial Bus specification. The USB-IF was formed to provide a support organization and forum for the advancement and adoption of Universal Serial Bus technology. The Forum facilitates the development of high-quality compatible USB peripherals (devices), and promotes the benefits of USB and the quality of products that have passed compliance testing. Some of the many activities that the USB-IF supports include:

- USB Compliance Workshops
- USB compliance test development
- www.usb.org website
- USB pavilions at Comdex, PC Expo, World PC Expo and other events
- Marketing programs and collateral materials, such as retail newsletters, retail salespeople training, store end-caps, etc.
- USB 2.0 Developer Conferences
- and many more...

Becoming a member of the USB-IF costs $2,500 for a one-year membership. An application for membership can be found at www.usb.org/developers/data/usbifapp.pdf.

USB 1.1 Specification
You should know that for every minute detail mentioned in this article about the USB specification, there are dozens of other details that have not been mentioned. The spec is daunting at best, and should only be read when suffering the worst bouts of insomnia. That said, the device drivers from FTDI do a splendid job of hiding the details of the specification, and this article will only cover the ones you are most likely to encounter when using the FT8U245AM. For those interested in reading USB specifications, they can be found at www.usb.org/developers/docs.html

The FT8U245AM is compliant with version 1.1 of the USB spec. By definition, version 1.1 allows for two speeds of communication: "Low Speed" which is 1.5 megabits per second, and "Full Speed" which is 12 megabits per second. The new USB specification 2.0 is rated at 480 megabits per second and is being termed "High Speed". The FT8U245AM communicates at up to 8 megabits per second, somewhat less than Full Speed due to the behind-the-scenes handling of all the details associated with USB communication.

The FT8U245AM is designed to interface directly with a 93C46 EEPROM for storing configurable parameters which include the PID (Product ID), VID (Vendor ID), device description and manufacturer name. Users of the FT8U245AM can use their own VID and PID or FTDI's. If you are planning to commercialize a product with a USB port, you must register your own PID and VID with the USB-IF. This registration is included with the $2,500 membership fee, or you can become a non-member USB-IF Logo Licensee for $1,500. Once registered, your VID and other configurable parameters can be written to the EEPROM using the program 232PROG.EXE which can be downloaded from FTDI's website. This program will also generate a serial number and write it to the EEPROM.

USB analyzers are available from a number of companies and tend to be rather pricey, but can be 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 very familiar with the USB Specification. At this level of understanding, you will probably want to write your own drivers.

Conclusion

The Universal Serial Bus is slowly changing the way we connect to Personal Computers. Several manufacturers are now selling USB based computers that do not have serial or parallel ports. The ability to connect and disconnect peripherals without having to shut down the computer as well as the elimination of ambiguity over where and how to connect a device can be extremely attractive to consumers. Legacy ports will be around for some time to come but, eventually, the lower cost of USB-based peripherals will choke them out of existence. It looks as if USB is far more than a passing fad--what better time to get acquainted?