UNIVERSAL SERIAL BUS – THE EASY WAY

By Don L. Powrie

06-18-2001

Several manufacturers make silicon for implementing a USB 1.1 compliant interface. Some with built in microcontrollers and some that are just the serial engine. They all have 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 without shutting down is very convenient. The one area of information that seems to be completely avoided is the topic of the inevitable device driver which USB devices simply will not operate without.

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 extremely tricky pieces of software. Only a very small percentage of programmers have the knowledge and experience necessary to write drivers. A driver that is not written well can result in the blue screen of death or complete system lockup – regardless of what operating system is running on your computer. 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 (www.ftdichip.com). After the virtual com port drivers are loaded, the application software merely has to open a COM port and read/write as though it were talking over standard RS-232. The drivers intercept the data that would otherwise go to the RS-232 port and feed it to the USB scheduler which then sends it on 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. Once a FT8U245AM is connected to your system and the drivers are loaded, you select which COM port you want via the System Properties page.

Data Rate

Version 1.1 of the USB specification outlines two data rates: “Low Speed” which transfers data at 1.5 megabits per second and "Full Speed" which transfers data at 12 megabits per second. The FT8U245AM transfers data at a maximum of 8 megabits per second, somewhat slower than full speed.

In order to achieve the 8 megabit data rate, the target microcontroller must be able to read and write data every 125 nanoseconds. This is a tall order for most microcontrollers unless they are running at a very high clock speed. Another limiting factor for USB devices in general is the 1 millisecond 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 is sent via the device driver to the USB scheduler. This scheduler puts a request onto the list of tasks for the USB host controller to perform. This will typically take at least 1 millisecond to execute because it will not pick up the new request until the next ' USB Frame' (the frame period is 1 millisecond).

There is therefore a sizeable overhead (depending on your required throughput) associated with moving the data from the application to the USB device. If data is sent 'byte at a time' by an application, this will severely limit the overall throughput of the system as a whole. The message here is always send data in packets if speed is critical.

DLL Based Driver
For some programming languages, communicating via RS-232 ports requires a communications library that handles the interrupts and provides an easy programming interface (Open, Read, Write, etc.). Using the communications library presents an extra step which some programmers, for simplicity's sake, might like to avoid.

FTDI recently released a new version of their drivers that are based on a DLL instead of the virtual com ports. To use the DLL, the application program

Implementation
The FT8U245AM is available from FTDI in surface-mount form as a 32 pin MQFP. Samples are currently available from the US distributor (www.saelig.com) in New York. Evaluation boards are available from www.dlpdesign.com (Figure 1).

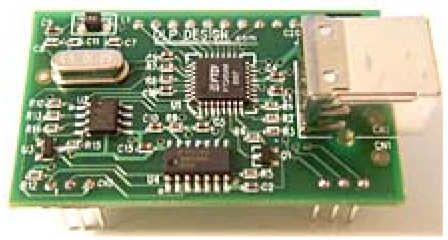
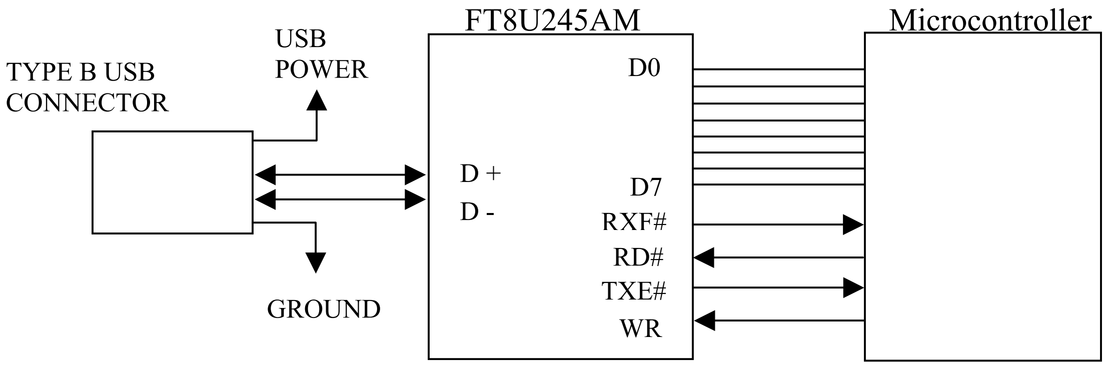


Figure 1 - DLP-USB1 from www.dlpdesign.com

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 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?