Overview

This document contains answers to a number of frequently asked questions about QuickUSB. This should only be used as a quick reference. Consult the QuickUSB User Guide and QuickUSB Application Notes for detailed information about QuickUSB.

Questions

1. What platforms is QuickUSB supported on?

QuickUSB is supported on x86 and x64 versions of: Windows XP, Windows Vista, Windows 7, Linux (kernel v2.6.25 and later), and Mac OS X (v10.6 and later). There is limited support for Linux running older than kernel v2.6.25—specifically, asynchronous IO may not be supported.

2. What data rates can I reach with QuickUSB?

The data rates you can achieve with QuickUSB depend on the architecture and implementation of a given design. Typically, a well-designed application using a fast IO Model and large data request sizes can achieve 20 – 30 MB/s. Some designs can perform at rates greater than 30 MB/s. Consult QuickUSB Application Note AN105 – 'Maximizing Data Throughput with QuickUSB' for more information.

3. Are there licensing fees or royalties when commercially selling products using QuickUSB?

No. When you purchase a QuickUSB Module you have purchased the module hardware along with one license of QuickUSB firmware for private or commercial use. We offer two firmware licensing options if you decide to design the QuickUSB circuit into custom hardware. They are:

a. Build the circuit into your own hardware and purchase pre-programmed EEPROMs from us (QuickUSB ChipPack EEPROMs), or
b. Build the circuit into your won hardware, supply you own EEPROMs, and electronically program them over the web (QuickUSB iChipPack) using our QuickUSB Programmer software.

There are no additional licensing fees or royalties beyond the cost of a QuickUSB iChipPack or QuickUSB ChipPack EEPROM.

4. How do I program my FPGA in order to talk to QuickUSB?

We provide sample FPGA Verilog code and Quartus II projects with the QuickUSB Evaluation Board Library that demonstrates how to interface QuickUSB with an FPGA. Additionally, there is a large amount of information about interfacing to a FPGA in the QuickUSB User Guide.

5. I need to integrate an USB 2.0 connection with the Altera DE2-115 FPGA board. Can I do it with a QuickUSB module?

Yes. To use QuickUSB with an Altera Development Board you could use, for example, a QuickUSB Module with a QuickUSB Adapter Board and wire the QuickUSB Module to the development board. The DE2-115 has a convenient 40-pin connector that could be used for that purpose.

6. Can I interface to QuickUSB with MATLAB?

Yes. To call QuickUSB API functions from within MatLab, you must use the *loadlibrary* and *unloadlibrary* functions to first load and unload the QuickUSB Library. Once the library is loaded, you may make calls to functions using the *calllib* function. The QuickUSB Library includes a sample that demonstrates how you can create an interface class to make calls to QuickUSB API functions and how to use that interface class to actually perform QuickUSB function calls.

For more information, refer to:

- http://www.mathworks.com/help/techdoc/ref/loadlibrary.html
- http://www.mathworks.com/help/techdoc/ref/calllib.html
- http://www.mathworks.com/help/techdoc/ref/unloadlibrary.html

7. Can I interface to QuickUSB with LabVIEW?

Yes, there are a few ways to interface the QuickUSB library to LabVIEW. They are as follows:

a. Have LabVIEW use the QuickUSB .NET Assembly (Recommended)

To use the .NET assembly, in LabVIEW you create a 'constructor node' object and use its reference output as an input to 'invoke' objects to call the API functions. Each 'invoke' object takes an error as an input and has an error output. The error input is used to prevent the function call if the input error line indicates failure. That way if a function call fails, you can prevent subsequent function calls from being made. Note that you do not have to create any interface VIs to use the .NET assembly in LabVIEW.

For more information, refer to:

- http://zone.ni.com/reference/en-XX/help/371361E-01/lvhowto/create_net_object/

b. Have LabVIEW generate interface VIs to the QuickUSB DLL

The files located in the 'Shared Library Import' folder, installed with the QuickUSB Library typically at 'C:\Program Files\Bitwise Systems\QuickUsb\Library\LabVIEW\Shared Library Import', are the VIs interfaces create using the 'Shared Library Import' function. You may use these VIs as an interface to the QuickUSB DLL.

To regenerate the VI interfaces, select 'Tools->Import->Shard Library (.dll)' from within the LabVIEW environment. Next, select the 'Create VIs for a shared library' option then click 'Next'. Specify the paths to the QuickUSB DLL and header file, then click 'Next'. In the preprocessor definitions window enter '_WIN32' and then click 'Next'. LabVIEW will parse the QuickUSB header file and extract functions for which it can create a VI. Continue with the wizard to generate the interface VIs.

Note: LabVIEW currently is unable to properly determine the interfaces for API functions that have structures (such as QBULKSTREAM) or pointers to functions (such as callback functions and completion routines). Because of this, LabVIEW is unable to automatically generate interface methods to the EPCS, Firmware, Streaming, and new Bulk Async API functions.

For more information, refer to:

- http://zone.ni.com/devzone/cda/tut/p/id/2818

c. Create your own interface VIs

The files located in the 'User Contributed' folder, installed with the QuickUSB Library typically at 'C:\Program Files\Bitwise Systems\QuickUsb\Library\LabVIEW\User Contributed', are user-contributed files that can help you get started with creating your own interface VIs to QuickUSB from scratch.

For more information, refer to:

- http://zone.ni.com/reference/en-XX/help/371361D-01/glang/call_library_function/

8. Is it possible to connect multiple QuickUSB devices to a single computer and communicate with both?

The QuickUSB Library and Driver are able to handle multiple QuickUSB devices connected to the same computer. The devices are identified by a call to *QuickUsbFindModules()* as "QUSB-0", "QUSB-1", "QUSB-2", etc.

9. I have accidently corrupted or erased the EEPROM in my device and it will no longer enumerate. What can I do?

If your QuickUSB device refuses to enumerate at all, meaning that it is not listed in the QuickUSB Programmer and it does not show up in the Windows Device Manager under the USB section, then the device is in a broken state. To fix it, the EEPROM on the device needs to be removed and either erased, reprogrammed, or replaced with a blank EEPROM. For assistance in with this process, please contact QuickUSB Support at support@quickusb.com.

10. How do I power my circuit from QuickUSB?

USB supplies 500 mA of unregulated +5V power. Of that power, you may safely use up to 400 mA to power other electronics when using QuickUSB. Please refer to the QuickUSB User Guide and relevant sections from Application Note AN103 – 'How to Successfully Design the QuickUSB Circuit Into a Custom PCB' about bus-powered design requirements.

11. How long does it take to change the state of a GPIO pin on the QuickUSB Module?

The time it takes to change the state of a GPIO pin is largely controlled by two factors: Systems latencies and USB transaction speed. System latencies originate from the time is takes to issue the QuickUSB API call and have it make its way down the USB driver stack. USB transaction speeds are the amount of time it takes for the request to physically transmit over USB. For a High-Speed connection, the time needed to change the state of a GPIO pin is approximately 125 microseconds plus system delays, while for a Full-Speed connection the time is approximately 1 millisecond plus system delays. The time the QuickUSB device takes to execute firmware in order to change the pin state may be considered negligible in comparison to the sum of system latencies and USB transaction times.

Bitwise Systems
6489 Calle Real, Suite E
Goleta, CA 93117
V (805) 683-6469
F (805) 683-4833
info@bitwisesys.com