

Optimizing Data Throughput Over USB

Introduction

This note provides guidelines for achieving optimal performance over USB from D2XX applications.

Transfer Sizes

The maximum transfer size over USB is setup once and for all when a device is initialized. An application can change the current USB transfer size to a value not exceeding the maximum USB transfer size. The best transfer size for an application depends on its type. For example, an application that handles streaming data will run more efficiently with a large transfer size, whereas a smaller transfer size may be more suited to an application that handles small amounts of data.

Driver Operation

In the D2XX device driver, the maximum USB transfer size is not configurable and is set at 64Kbytes. This is necessary in order to support USB2.0 controllers: on USB2.0, we have found that transfer sizes greater than 64K fail catastrophically.

Current USB transfer size can be changed using the function *FT_SetUSBParameters*. The change comes into effect immediately, and any data that was held in the driver at the time of the change is lost.

Note that, currently, *FT_SetUSBParameters* supports transfer size changes for the IN endpoint only, and the remainder of this note is restricted to consideration of read requests.

When a device is opened, the driver allocates two buffers: a USB transfer buffer of maximum USB transfer size bytes, and a read buffer whose size is based on, but not the same as, the maximum USB transfer size. The current USB transfer size is set by default to 4Kbytes, but an application may setup the current USB transfer size anytime after a device has been opened. The driver performs USB requests of any size up to the current USB transfer size, then copies the data to the read buffer. The actual USB request size is determined by the space that is currently available in the read buffer. If the read buffer becomes full (i.e. there is not enough space in the read buffer to hold data returned from a USB request), the driver will stop issuing USB requests until space becomes available.

Data is removed from the read buffer when the application performs a read request. So the rate at which the application issues read requests has a direct bearing on the driver's ability to keep issuing USB requests. USB requests will continue to be issued as long as the application performs enough read requests to ensure that the read buffer does not become full.

Optimizing Performance

The question of how to setup an application for optimized data throughput is complicated by the format of the data that is transferred over USB. Data is transferred over USB in a series of 64 byte packets. Each packet contains two bytes that are reserved by FTDI and 62 bytes of actual data. So a 4Kbyte USB transaction contains 3968 bytes of actual data, and a 64Kbyte transaction contains 63488 bytes. In general, an x byte USB transaction yields $x - ((x/64)*2)$ bytes of actual data.

Optimized throughput occurs when the maximum amount of data is transferred in the least possible number of USB transactions. The necessary condition is that the application's request size must be matched to the current USB request size. More precisely, optimal throughput is achieved when the application request size is a multiple of the number of actual data bytes contained in the current USB transfer size. For example, using a current USB transfer size of 4Kbytes, optimal throughput is achieved with application request sizes that are multiples of 3968 bytes.

Maximum throughput is achieved using a current USB transfer size of 64Kbytes, and application requests that are multiples of 63488 bytes.