Data Rates and Flow Control for USB to RS232

Background

This application note describes the major architecture differences that need to be considered by both software and hardware designers when changing from a traditional RS232 based solution, to one that uses the FT8U232 USB to serial device.

Data Transfer Comparison

The traditional RS232 COM port in a PC plugs directly into the mother board and hooks to one of the system interrupts. When a character is transmitted or received (depending if FIFO's are used) the CPU is interrupted and executes a data handling routine. This means that a user could be confident that, given a particular Baud rate and Data rate, data could be transferred without any need for flow control. The hardware interrupt ensured that the transfer would get serviced.

With a USB device data is transferred in packets. If data is to be sent from the PC, a packet is built up by the device driver and sent 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.

Data is received from USB to the PC by a polling method. The driver requests a certain amount of data from the USB scheduler. This will be transferred in multiples of 64 bytes, which is the maximum 'bulk packet size' on USB. The host controller reads data from the device until either (a) a packet shorter than 64 bytes is received or (b) the requested data length is reached. The device driver will request packet sizes between 64 Bytes and 4 KBytes. The size of the packet will affect the performance and is dependent on the data rate.

For very high speed, the largest packet size is needed. For 'real-time' applications that are transferring audio data at 115200 Baud, for example, then the smallest packet is desirable, otherwise the device will be holding up 4k of data at a time. The latest driver release, scheduled for mid-January 2000, will automatically vary the requested packet size depending on the selected baud rate.

In order for single characters to get passed through, there is a timer inside the FT8U232 chip. It is used for data from RS232 to the PC. If the buffer internal to the device has 64 bytes in it, then it is setup to be sent the next time the host does a 'Read' request. If the buffer is not empty, but does not have 64 bytes in it, then it will be setup for sending after the timer has expired.

A timer value of 32 milliseconds was chosen so that we could make advantage of 64 byte packets to fill large buffers when in high speed mode, as well as letting single characters through. This means that it will take 32 milliseconds to receive an individual character over and above the transfer time on RS232. For large amounts of data, at high data rates, the timer is not required. It may be used for the last packet of a block if the packet size ends up as less than 64 bytes.

The first 2 bytes of every packet are used as status bytes for the driver. This status is sent every 32 milliseconds, even when no data is present.

It must be stressed that application programs should send or receive data using buffers and not individual characters.

Event Character

Event characters can be used to access the data more quickly. Applications that access the internet, for example, can program up the event character as $7E. Data is then sent and received in packets that have $7E at the start and end of the packet. When event character matching is enabled and the hardware detects that the current character is the same as the event character then the hardware will immediately send the buffer.

Flow Control

There are 4 methods of flow control that can be programmed for the FT8U232 device.

1. None - this may result in data loss at high speeds

2. RTS/CTS - 2 wire handshake. The device will transmit if CTS is active and will drop RTS if it cannot receive any more.

3. DTR/DSR - 2 wire handshake. The device will transmit if DSR is active and will drop DTR if it cannot receive any more.

4. XON/XOFF - flow control is achieved by sending or receiving special characters. One is XON (transmit on) the other is XOFF (transmit off). They are individually programmable to any value.

Flow control is encouraged because we cannot ensure that we will always be scheduled. The chip can buffer up to 384 bytes of data. Windows can 'starve' the driver program of time if it is doing other things such as moving an application around the screen with the mouse. This results in a lot of graphics activity and data loss can occur if receiving data at 115200 baud (for example) with no handshaking. If the data rate is low or data loss is acceptable then flow control may be omitted.