Application Note

AN232B-04

Data Throughput, Latency and Handshaking

Version 1.2

Issue Date: 18-03-2024

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 FT232R, FT232RN or FT-X (excluding FT240X) USB to serial interface devices.

Future Technology Devices International Limited (FTDI)
Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777   Fax: + 44 (0) 141 429 2758
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

1.1 Data Throughput, Latency and Handshaking

The Universal Serial Bus may be new to some users and developers. This application note tries to describe 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 FT232R, FT232RN or FT-X (excluding FT240X) USB to serial interface devices. Much of the information in this application note also applies to the FT245R, FT245RN, and FT240X to parallel FIFO interface devices.

2 Data Transfer

2.1 The Need for Handshaking

USB data transfer is prone to delays that do not normally appear in systems that have been used to transferring data using interrupts. The original COM ports of a PC were directly connected to the motherboard and were interrupt driven. When a character was transmitted or received (depending if FIFO's are used) the CPU would be interrupted and go to a routine to handle the data. This meant that a user could be reasonably certain that, given a particular baud rate and data rate, the transfer of data could be achieved without any real need for flow control. The hardware interrupt ensured that the request would get serviced. Therefore, data could be transferred without using handshaking and still arrive at the PC without data loss.

2.2 Data Transfer Comparison

USB does not transfer data using interrupts. It uses a scheduled system and as a result, there can be periods when the USB request does not get scheduled and, if handshaking is not used, data loss will occur. An example of scheduling delays can be seen if an open application is dragged around using the mouse.

For a USB device, data transfer is done in packets. If data is to be sent from the PC, then a packet of data is built up by the device driver and sent to the USB scheduler. This scheduler puts the 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). Therefore, there is a sizable overhead (depending on your required throughput) associated with moving the data from the application to the USB device. If data were sent 'a byte at a time' by an application, this would severely limit the overall throughput of the system.

2.3 Continuous Data – Smoothing the Lumps

Data is received from USB to the PC by a polling method. The driver will request a certain amount of data from the USB scheduler. This is done in multiples of 64 bytes. The 'bulk packet size' on USB is a maximum of 64 bytes. The host controller will read 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 4Kbytes. 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, the smallest packet possible is desirable, otherwise the device will be holding up 4k of data at a time. This can give the effect of 'jerky' data transfer if the USB request size is too large and the data rate too low (relatively).

3 Buffers and the Latency Timer

3.1 Small Amounts of Data and End of Buffer Conditions

When transferring data from an FTDI USB-Serial or USB-FIFO IC device to the PC, the device will send the data given one of the following conditions:

1. The buffer is full (64 bytes made up of 2 status bytes and 62 user bytes).
2. One of the RS232 status lines has changed (USB-Serial chips only). A change of level (high or low) on CTS# / DSR# / DCD# or RI# will cause it to pass back the current buffer even though it may be empty or have less than 64 bytes in it.
3. An event character had been enabled and was detected in the incoming data stream.
4. A timer integral to the chip has timed out. There is a timer (latency timer) in the FT2xxR, FT2xxRN and FT-X chips that measures the time since data was last sent to the PC. The default value of the timer is set to 16 milliseconds. Every time data is sent back to the PC the timer is reset. If it times-out, then the chip will send back the 2 status bytes and any data that is held in the buffer.

From this it can be seen that small amounts of data (or the end of large amounts of data), will be subject to 16 milliseconds delay when transferring into the PC. This delay should be taken along with the delays associated with the USB request size as mentioned in the previous section. The timer value 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. Since the value chosen for the latency timer is 16 milliseconds, this means that it will take 16 milliseconds to receive an individual character, over and above the transfer time on serial or parallel link.

For large amounts of data, at high data rates, the timer will not be used. It may be used to send the last packet of a block if the final packet size works out to be less than 64 bytes. The first 2 bytes of every packet are used as status bytes for the driver. This status is sent every 16 milliseconds, even when no data is present in the device.

A worst-case condition could occur when 62 bytes of data are received in 16 milliseconds. This would not cause a time-out but would send the 64 bytes (2 status + 62 user data bytes) back to USB every 16 milliseconds. When the USBD system driver receives the 64 bytes it would hold on to them and request another 'IN' transaction. This would be completed another 16 milliseconds later and so on until USBD gets all the 4K of data required. The overall time would be (4096 / 64) * 16 milliseconds = 1.024 seconds between data packets being received by the application. To stop the data arriving in 4K packets, it should be requested in smaller amounts. A short packet (< 64 bytes) will of course cause the data to pass from USBD back to the FTDI driver for use by the application.

For application programmers it must be stressed that data should be sent or received using buffers and **not** individual characters.

3.2 Adjusting the Receive Buffer Latency Timer

FTDI's R, C and BM series chips allow the latency timer to be changed from 16 milliseconds to any value from 1 to 255 milliseconds, in 1 millisecond increments. When using the FTDI Virtual COM Port driver the latency timer can be set in the port properties page.

In Windows, the port properties page is accessed via the Control Panel > System > Device Manager. For Windows 2000 and XP, the initial value of the latency timer can also be pre-configured in `ftdiport.inf` by changing the value of the last number in the following line:

```
[FtdiPort.NT.HW.AddReg]
HKR,,"LatencyTimer",0x00010001,16
```

where again, 16 milliseconds is the default value.

When using FTDI's D2XX direct driver the function **`FT_SetLatencyTimer`** can be used to adjust the value of the latency timer.

3.3 Effect of USB Buffer Size and the Latency Timer on Data Throughput

An effect that is not immediately obvious is the way the size of the USB total packet request has on the smoothness of data flow. When a read request is sent to USB, the USB host controller will continue to read 64 byte packets until one of the following conditions is met:

1. It has read the requested size (default is 4Kbytes).
2. It has received a packet shorter than 64 bytes from the chip.
3. It has been cancelled.

While the host controller is waiting for one of the above conditions to occur, NO data is received by our driver and hence the user's application. The data, if there is any, is only finally transferred after one of the above conditions has occurred.

Normally, condition 3 will not occur so we will look at cases 1 and 2. If 64 byte packets are continually sent back to the host, then it will continue to read the data to match the block size requested before it sends the block back to the driver. If a small amount of data is sent, or the data is sent slowly, then the latency timer will take over and send a short packet back to the host which will terminate the read request. The data that has been read so far is then passed on to the users' application via the FTDI driver. This shows a relationship between the latency timer, the data rate and when the data will become available to the user. A condition can occur where if data is passed into the FTDI chip at such a rate as to avoid the latency timer timing out, it can take a long time between receiving data blocks. This occurs because the host controller will see 64 byte packets at the point just before the end of the latency period and will therefore continue to read the data until it reaches the block size before it is passed back to the user's application.

The rate that causes this will be:

62 / Latency Timer bytes/Second

(2 bytes per 64 byte packets are used for status)

For the default values:

62 / 0.016 ~= 3875 bytes /second ~= 38.75 KBaud

Therefore, if data is received at a rate of 3875 bytes per second (38.75KBaud) or faster, then the data will be subject to delays based on the requested USB block length. If data is received at a slower rate, then there will be less than 62 bytes (64 including our 2 status bytes) available after 16 milliseconds. Therefore, a short packet will occur, thus terminating the USB request and passing the data back. At the limit condition of 38.75KBaud it will take approximately 1.06 seconds between data buffers into the users' application (assuming a 4Kbyte USB block request buffer size).

To get around this you can either increase the latency timer or reduce the USB block request. Reducing the USB block request is the preferred method though a balance between the 2 may be sought for optimum system response.

3.4 Adjusting the USB Transfer Size

Again, when using the FTDI Virtual COM Port drivers, the USB Transfer (buffer) size can be set in the port properties page. The initial buffer size is calculated from entries in the `ftdiport.inf` file - with the size of buffer allocated being equal to the `.inf` entry plus 1 multiplied by 64 (bytes).

So, 0 is 64 bytes, and 3F is (63+1) *64 = 4096.

There are two entries in the INF file - the first one is the transmit buffer and the second is the receive buffer.

```
[FtdiPort.NT.HW.AddReg]
HKR,,ConfigData,1,01,00,3F,3F,10,27,88,13,C4,09,E2,04,71,02,38,41,9c,80,4E,C0,
34,00, 1A,00,0D,00,06,40,03,80,00,00,00,00
```

In the example above, the two 3F's are the entries in question, with this line being set for 4K byte buffer size operation and the below example being set for 64 byte buffer size operation.

```
[FtdiPort.NT.HW.AddReg]
HKR,,ConfigData,1,01,00,00,00,10,27,88,13,C4,09,E2,04,71,02,38,41,9c,80,4E,C0,
34,00, 1A,00,0D,00,06,40,03,80,00,00,00,00
```

When using FTDI's D2XX direct driver the function **`FT_SetUSBParameters`** can be used to adjust the size of the USB block requested, as indicated in AN232B-03 Optimizing D2XX Data Throughput application note, see https://ftdichip.com/document/application-notes/.

4 Events and Flow Control

4.1 Event Characters

Event characters can be used with FT2xxR, FT2xxRN, or FT-X series devices. If the event character is enabled and it is detected in the data stream, then the contents of the devices buffer is sent immediately. The event character is not stripped out of the data stream by the device or by the drivers, it is up to the application to remove it.

Event characters may be turned on and off depending on whether large amounts of random data or small command sequences are to be sent. The event character will not work if it is the first character in the buffer. It needs to be the second or higher. The reason for this being applications that use the Internet for example, will program the event character as '$7E'. All the data is then sent and received in packets that have '$7E' at the start and at the end of the packet. To maximise throughput and to avoid a packet with only the starting '$7E' in it, the event character does not trigger on the first position.

4.2 Flushing the Receive Buffer Using the Modem Status Lines

Flow control can be used by the FT2xxR, FT2xxRN, and FT-X series devices to flush the buffer in the chip. Changing one of the modem status lines will do this. The modem status lines can be controlled by an external device or from the host PC itself. If an unused output line (DTR) is connected to one of the unused inputs (DSR), then if the DTR line is changed by the application program from low to high or high to low, this will cause a change on DSR and make it flush the buffer.

4.3 Flow Control

The FT245R, FT245RN and FT240X chips use their own handshaking as an integral part of its design, by proper use of the TXE# line. The FT232R, FT232RN, and FT-X series (excluding FT240X) chips can use RTS/CTS, DTR/DSR hardware or XOn/XOff software handshaking. It is highly recommended that some form of handshaking be used.

There are 4 methods of flow control that can be programmed for the FT232R, FT232RN, and FT-X (excluding FT240X) devices.

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 done by sending or receiving special characters. One is XOn (transmit on) the other is XOff (transmit off). They are individually programmable to any value.

It is strongly encouraged that flow control is used because it is impossible to ensure that the FTDI driver 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. The most obvious example of this is moving an application around the screen with the mouse by grabbing its task bar. This will result in a lot of graphics activity and data loss will occur if receiving data at 115200 baud (as an example) with no handshaking. If the data rate is low or data loss is acceptable then flow control may be omitted.

5 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited (UK)
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales)	sales1@ftdichip.com
E-mail (Support)	support1@ftdichip.com
E-mail (General Enquiries)	admin1@ftdichip.com

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited (USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-mail (Sales)	us.sales@ftdichip.com
E-mail (Support)	us.support@ftdichip.com
E-mail (General Enquiries)	us.admin@ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan, R.O.C.
Tel: +886 (0) 2 8797 1330
Fax: +886 (0) 2 8751 9737

E-mail (Sales)	tw.sales1@ftdichip.com
E-mail (Support)	tw.support1@ftdichip.com
E-mail (General Enquiries)	tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
China
Tel: +86 (21) 62351596
Fax: +86 (21) 62351595

E-mail (Sales)	cn.sales@ftdichip.com
E-mail (Support)	cn.support@ftdichip.com
E-mail (General Enquiries)	cn.admin@ftdichip.com

Web Site

http://ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A – References

Document and Web References

AN232B-03 Optimizing D2XX Data Throughput

Acronyms and Abbreviations

Terms	Description
USB	Universal Serial Bus

Appendix B – List of Tables and Figures

List of Tables

N/A

List of Figures

N/A

Appendix C – Revision History

Document Title:	AN232B-04 Data Throughput, Latency and Handshaking
Document Reference No.:	FT_001567
Clearance No.:	FTDI#595
Product Page:	https://ftdichip.com/
Document Feedback:	Send Feedback

Revision	Changes	Date
1.0	Initial Release	03-2004
1.1	Format updated. Various corrections throughout. Release version on the web.	02-2006
1.2	Removed EOL ICs and added FT-X series and RN chips.	18-03-2024