An Analysis of Throughput Characteristics of Universal Serial Bus

John Garney, Media and Interconnect Technology, Intel Architecture Labs

Abstract

Universal Serial Bus (USB) is a new personal computer (PC) interconnect that can support simultaneous attachment of multiple devices. Developers of USB devices have initially had concerns about guaranteed throughput performance compared to previous popular single device PC connections such as RS232 serial ports or Centronics parallel ports. This paper presents a simple detailed analysis of USB throughput characteristics. That analysis is then used to model several examples of single USB device throughput requirements and a range of representative multiple device USB configurations. These examples demonstrate the broad range of potential USB devices that might possibly be implemented and expected to reliably work when added to an existing USB configuration. The reader is assumed to be familiar with the USB system architecture.

Introduction

Universal Serial Bus has features that allow simultaneously attaching and using multiple devices on the same bus. Older connections like RS232 COMx serial ports and parallel printer LPTx ports can only be connected to a single device at a time. USB also allows these devices to be attached and removed while the computer system is running and without requiring a reboot to use a newly attached device.

The 12Mb/s full speed (FS) bandwidth of USB allows the creation of very exciting low to medium speed devices. USB is intended for devices in the 8Mb/s and below range. USB also provides a lower cost, reduced feature mode of operation for low speed devices. This mode uses an $1/8^{th}$ speed clock resulting in a 1.5Mb/s low speed (LS) bandwidth. Other interconnects will provide support for higher speed devices.

USB also has features to support isochronous devices like telephones. Game and telephony device developers are particularly interested in these features. Therefore, a wide variety of new and existing peripherals may be developed that utilize USB. Developers need to ensure that their devices will operate well in combination with other devices so that end user expectations are met.

Developers of USB devices can determine specific details about USB from the USB Specification [1]. However, the USB specification presents raw data about the throughput performance features of the bus, but doesn't present them in a fashion that allows a developer to easily determine a reasonable throughput design target for a specific USB device.

This paper uses the raw data from the USB specification to give a developer a better look at how device data rates appear on the bus and also several example configurations. First some of the key characteristics of USB are described to understand differences as compared to other common PC cabled buses. A spreadsheet is then developed to model the throughput requirements of a single device given the raw data rate and USB transfer type(s) needed to support the device. This spreadsheet is then used with estimated data for several example devices. Finally, the estimated data for these example devices are used to construct a range of USB configurations to illustrate the remaining bandwidth that can be used for additional devices.

The tables and figures included in this paper are extracted from a spreadsheet built for the analysis.

Key USB Throughput Characteristics

Critical USB requirements are imposed on USB host controllers in order to support isochronous data transfers. One requirement is that transactions on the bus are always framed into 1ms quanta. Therefore, this analytic model focuses on the number of "byte times per frame (BTF)", e.g. it looks at the opportunities to transmit bytes in a 1ms time period even though the bus is a bit serial bus. Table 1 shows USB's raw bus speed, 12MHz. This translates into the time to transmit 1500 bytes during a 1ms USB frame.

Bus Characteristics:			
Bus Speed		Bit time	Byte time
1.2E+07		8.33E-08	7E-07
Max BW	Raw BW	adj %	
1500000	1500000	1	
Frame time			
0.001 sec.			
Max per frame		Max Frame w/ Overhead	
1500 bytes		*1308*	
Protocol Overhead		Bit Stuff %	Max BS
14		**0.1667**	218

Table 1 - USB Bus Characteristics

Bus Overhead

There are also several USB attributes that reduce the actual number of bytes that can be used by other devices during a frame. These attributes manifest as overhead of the bus when viewed from the desired throughput requirements of a device. Specific sources of overhead include: packet organization, Start of Frame (SOF), end of frame (EOF), clock adjustment, time consumed to poll hubs, and time reserved for control transfers.

An SOF packet is transmitted over the bus to identify the beginning of every USB frame. This packet consists of 8 bits of SYNC, an 8 bit packet identifier (PID), an 11 bit frame count, a 5 bit CRC, 3 bits of end of packet (EOP) and approximately 13 bits of inter packet spacing. This takes approximately 6 BTF to transmit.

In order to preserve the integrity of 1ms framing and to guarantee the ability to generate an SOF, there is also an EOF "guardband" at the end of each frame that precedes the next frame's SOF. Hubs use this time to monitor the state of the bus and disable devices that transmitting when they shouldn't be. This EOF time accounts for approximately 2 BTF.

USB allows its specific frame size to be adjusted so that SOFs can be slaved to an external device's clock such as a Public Switched Telephony Network(PSTN) clock. This can result in the nominal 12000 bits per frame varying by +/- 16 bits. This means that a frame may actually be up to 2 BTF shorter.

USB depends on standardized devices called hubs that support its hot plugging features and multiple connectors. Hubs must be polled periodically to determine if devices have been attached to or removed from the bus. This hub polling can occur with low frequency, e.g. 1 poll every 255ms. A hub poll requires an interrupt transfer on the bus. As will be described later for Table 2, this interrupt transfer requires 14 bytes of transaction protocol overhead plus 2 bytes of data when there is a change. When no change is being reported by a hub, 8 bytes time are saved since a negative acknowledgment (NAK) handshake packet instead of a data and handshake packet is transmitted.

The last specific overhead is to provide time for a control transfer in each frame. The specification requires that 10% of each frame be usable for any requested control transfers. Time for control transfers must be available to allow recognition and configuration of newly attached devices. 10% of a frame is 150 BTF, however, for simplicity, this analysis assumes that there is sufficient time for one low speed transaction. A low speed transaction of 8 bytes (the maximum data payload allowed for low speed) is 162 Full Speed equivalent BTF; 98 BTF overhead + 64 BTF for the data payload (8 LS bytes * 8x bit time due to $1/8^{th}$ clock).

Estimated Transaction Protocol Overhead BTF:	
interpacket delay (included in overheads)	2
protocol overhead for FS non Isochronous	14
protocol overhead for Isochronous output	10
protocol overhead for Isochronous input	11
protocol overhead for LS	98
EOF guard band FS	6
EOF guard band LS	23

Table 2 - Estimated Protocol Overhead Byte Times per Frame

Table 2 shows the estimated overhead BTF per transaction for various USB defined transfer types[1]. These are simply computed BTF equivalents of the nanosecond times reported in Section 5.9.3 of the USB Specification. The "non Iso" entry is applicable for control, bulk and interrupt transfer types.

[1] The estimated transaction protocol overheads that are shown assume minimal host system and host controller implementation delays. For systems that are less efficient then the estimate system,, additional delays can be introduced due to processing time required or latency imposed to access memory. The values shown are only estimates, not actual values measured on real hardware.

The "Max Frame w/ Overhead" entry in Table 1 shows the 1308 remaining available BTF for other device use after these overheads are subtracted from the raw bytes times per frame. The actual overhead present in a given frame can be less than identified here since there may be less clock adjustment, no hub polls in this frame and smaller or nonexistent control transfers. This available BTF can be considered a lower bound on bus time that can be allocated to a newly attached device.

Bit Stuffing Overhead

The USB bus is conceptually a single half duplex wire that carries clock and data on the same wire. NRZI encoding of the bit stream is specified for the physical link. USB uses a feature called bit stuffing to ensure the presence of sufficient signal transitions for clock recovery.

USB bit stuffing consists of inserting an additional 0 bit in the physical bit stream on the bus after 6 consecutive 1 bits. Therefore, bit stuffing can consume up to 16.67% additional bus time to move an amount of data. Also, the actual additional time required is data dependent since bits are only added after 6 consecutive 1's are present in the physical data stream. Since a specific data stream will not in general be known, an estimate must be used for bit stuffing overhead.

The worst case bit stuffing is 16.67%, but Intel's internal simulations have indicated that 0.8% bit stuffing may be typical for random bit streams. Given the wide range of possible bit stuffing overheads, the examples described later in this analysis will use multiple bit stuffing estimates.

Table 1 is shown using the worst case (16.67%) bit stuffing overhead. This bit stuffing is computed over the remaining data bytes (1308) and results in 218 BTF of stuffed bits. This may be too high since many of the protocol overhead bytes are constructed so as to require no bit stuffing. However since it not possible to *a priori* determine the specific protocol overhead present in a given frame, this number is used as the worst case upper bound bit stuffing overhead.

The bold entries at the bottom of Table 1 show where the input values for protocol overhead and estimated bit stuffing percentage are supplied to the spreadsheet when modeling a USB device transfer.

As can be seen, the transfer type that a specific device uses affects the bus time required for protocol overhead. This protocol overhead must be added to the bus time required to transmit the device data payload. For example, consider transmission of a full speed 8 byte data payload via an output control transfer. A control transfer consists of 3 stages, each being a transaction. A control transfer requires an 8 byte data payload setup stage transaction (14 overhead + 8 setup data payload bytes), the desired 8 byte data payload data stage transaction (14 overhead + 8 data payload bytes), and a zero byte status stage (14 overhead bytes). This is a total of 58 BTF to transmit the transfer.

In comparison, transferring the same 8 data payload

Possible Bandwidth Delivery Choices/Impact									
			Best (no bit stuff,etc.)			Worst (w/ bitstuff, ovrhd)		
payload size	max BW Bytes/sec	%max BW	max trans/f	Bytes/fram Remaining	Bytes/frame useful data	max trans/f	Bytes/fra remaining	Bytes/ Frame
1	100000	0.07	100	0	100	72	10	72
2	198000	0.13	93	12	186	68	2	136
4	338000	0.23	83	6	332	60	10	240
8	548000	0.37	68	4	544	49	12	392
16	800000	0.53	50	0	800	36	10	576
32	1052000	0.70	32	28	1038	23	32	754
64	1234000	0.82	19	18	1220	13	76	894
128	1360000	0.91	10	80	1346	7	96	978
256	1430000	0.95	5	150	1416	4	10	1024
max	1500000				1500			

Table 3 - Payload Sizes vs. Number of Transactions per Frame

Device Desired Bandwidth:			Worst	Max desired BW	Max Avail
bits/sec	bytes/sec	bytes/frame	bit stuff	w/ worst bitstuff	Data BW
6291456	786432	*787*	131	*918*	*1090*

Table 4 - Desired Device Bandwidth Calculations

bytes via an isochronous transfer takes only 18 BTF. That is 10 bytes of protocol overhead and 8 bytes of data payload. Therefore the designer of a device should carefully choose the transfer types used to communicate with the device to be as efficient as possible.

A USB device can be composed of several simultaneously used endpoints where each endpoint can have a different transfer type with its corresponding bandwidth requirements. This analysis assumes simple devices with single endpoints for each data transfer. The same spreadsheet can be used for more complex devices.

Table 2 can be used as a reference to select the appropriate value to fill in the protocol overhead field of Table 1. As shown in the example of Table 7 below, the values for the protocol overhead fields in Table 1 may need to be used for multiple transactions, e.g. for control transfers in particular. The protocol overheads don't include the data payload size. For example, a control transfer would be 50 BTF of overhead (not 58 BTF as indicated above since the desired 8 byte data payload is not part of the overhead).

Guaranteed vs. Non-Guaranteed Transfers

There are two classes of transfers from a bandwidth allocation perspective: guaranteed and non-guaranteed. Guaranteed transfers must have bandwidth allocated on the bus that ensures sufficient time for their transfer. Non guaranteed transfers have no bandwidth allocated for their use: these transfers will make use of bandwidth that is not being used for other purposes. Isochronous and interrupt transfer types are guaranteed. Control and bulk transfers are non-guaranteed.

Guaranteed transfers need to assume worst case bit stuffing in calculating their bandwidth requirements to ensure they will not fail simply due to lack of time due to addition of bits stuffed.

Non-guaranteed transfers can assume random data stream bit stuffing since a transfer will be retried if it doesn't complete due to lack of time.

General Bus Throughput

Table 3 shows how overall bus throughput varies as different data payload sizes are used to move data over the bus. Power of 2 data payload sizes from 1 to 256 are used to illustrate the effects of transaction protocol overhead. This table is generated based on the values supplied in Table 1. The analysis assumes that a frame is packed with transactions of the indicated size. Since isochronous and interrupt transfers have at most one transaction per frame, this table is of most use for analyzing control and bulk transfers.

The first column identifies the data payload size used. The second column shows the maximum data payload throughput in bytes/sec that can be delivered for this case. Column three shows the percentage of theoretical maximum throughput this case achieves.

The next two sets of three columns show results for two bit stuffing cases: no bit stuffing or frame overhead and worst case bit stuffing and full overhead. The first column of each case shows the maximum number of transactions that will fit in a 1ms frame. The second column shows the bytes remaining that is smaller than another full transaction. The third column shows the full data payload bytes per frame carried in each frame.

For example, the row for an 8 byte data payload of Table 3 shows that for a 14 byte overhead transfer type with no bit stuffing (best case), 68 transactions can be transmitted within a frame with 4 bytes per frame left over. Worst case bit stuffing reduces this to 49

			Best (no bit stuff,etc.)			Worst (w/ bitstuff, ovrhd)		
payload size	max BW	%max BW	max trans	Bytes Remaining	Bytes/frame useful data	max trans	Bytes remaining	Bytes/ Frame
787	1486000	0.99	1	699	1472	1	289	1062

Table 5 - Isochronous Data Payload and Frame Impact

transactions per frame with 12 bytes left over.

Single Device Model Spreadsheet

Table 4 does some initial calculations for a single theoretical USB device. Its desired transfer data rate is specified in the first column in bits per second and converted to a raw bytes per frame in column 3. A worst case bit stuff bytes is calculated in the next column.

The 2nd to last column shows the worst case bit stuffed total bytes per frame required to carry the desired data rate without protocol overhead. The last column shows the maximum possible data payload bytes per frame after worst case bit stuffing bytes have been consumed. This value is simply computed from the maximum frame overhead of Table 1 minus the worst case bit stuff (also from Table 1). This value is not dependent on the desired bandwidth, but can be compared with the value computed from the desired bandwidth in the 2nd to last column. If the desired bytes per frame is larger than the maximum available, then there is no way to reliably carry the desired data rate over USB.

The specific transaction protocol overhead still needs to be added to the value in the 2nd to last column to complete modeling the data throughput requirements of this device. If this computed value is larger than the "Max Frame w/ Overhead" value in Table 1, there is also no way to reliably carry this desired data rate over USB.

Table 5 is similar to Table 3 and shows how the desired data rate affects a frame assuming the data rate is moved as one transaction. This table is most useful when large data rates are desired. Large data payloads are carried most efficiently via isochronous transfers for two reasons: 1) the protocol overhead is the least, and 2) the maximum transaction data payload size allowed greatly exceeds the sizes allowed for the other transfer types. This table may also be useful for analyzing interrupt transfers as long as the data payload desired is not larger than the maximum size allowed by USB specification. The explanation of the remaining columns is the same as for Table 3.

Table 6 is most useful for analyzing bulk and control transfers. Each row corresponds to a power of 2 data payload size. The first two columns of Table 6 show results similar to Table 3. The third column shows how many BTF are remaining in the frame for the indicated data payload size and desired data rate including the specified transaction protocol overhead. If the number is negative, then that data payload size and transfer type cannot be used to carry the desired data rate. The last column shows the percentage of overall available transactions of this size that this case consumes in terms of required number of transactions compared to best case number of transactions.

The last row corresponds to Table 5 and shows details if a single transaction of maximum required size is used.

For example, Table 6 shows that for a desired data payload of 787 bytes per frame (as specified in Table 4), a data payload size of at least 128 bytes is required to be able to carry this data rate over USB. The smaller data payload sizes have too much overhead and require more time per frame than possible to move the data. Using a payload size of 128 bytes results in 33 bytes per frame remaining. This compares to using one data payload of 787 bytes (last row) with 117 bytes remaining.

Transaction Organization Choices			
test trans per frame	extra bytes	rem B/frame	%avail trans used
787	0	-10887	7.8700
394	1	-5385	4.2366
197	1	-2627	2.3735
99	5	-1255	1.4559
50	13	-569	1.0000
25	13	-219	0.7813
13	45	-51	0.6842
7	109	33	0.7000
4	237	75	0.8000
			0.5243
1	0	117	1.0000

Table 6 - Data Payload Sizes and Frame Impact

Example USB Devices

The spreadsheet described by Tables 3-6 is now used to present estimated bandwidth requirements of several "example" USB devices. The performance of the "example" devices has been estimated by Intel based on data gathered from various sources. These example devices have been selected to demonstrate a range of typical bandwidth demands. The devices are:

- a stereo CD quality isochronous output device like speakers, 44.1K samples/second, 16 bits per sample. This device is estimated to require 180 data payload bytes per frame worst case. It would actually require that amount only every 10th frame since the exact number of bytes required to deliver 44100 samples per second is not evenly divisible by 1000 (1ms) and so varies between 176 and 180. 10 bytes of isochronous transaction protocol overhead is also required (giving 190 bytes per frame total).
- an MPEG2 6Mb/s isochronous video camera. The data payload is estimated to require 787 bytes per frame. An additional 11 bytes would be required for transaction protocol overhead.
- a low speed interrupt device like a mouse or keyboard or joystick. This example device assumes a single low speed interrupt transfer polled every 8th frame with an 8 byte data payload. The 8 byte low speed payload appears as 64 full speed bytes. Adding the 98 (equivalent full speed) byte low speed transaction protocol overhead gives 162 bytes per frame.
- an 2B+D ISDN isochronous telephony device (only considering the main data streams and ignoring any additional synchronization or control data streams). This device is modeled as 2 isochronous data streams. Each B channel is 64Kb/s bidirectional, while the D channel is 16Kb/s bidirectional. This is 63 bytes of data payload with an input (11 byte) and output (10 byte) isochronous transaction protocol overhead included. This assumes that the D channel and the 2 B channels are all combined in only 2 USB isochronous streams. Other implementations with different bandwidth requirements are also possible.

Table 7 captures the results of using the spreadsheet for these devices. The values in the first four columns show the data payloads required including the transaction protocol overhead. Bit stuffing overhead is not included in these figures so that it can be varied in the multiple device configuration examples presented in the next section.

Example Bus Configurations

Since USB supports multiple simultaneous devices, a representative bus configuration must be used in order to estimate and analyze what can be expected when adding a new device to an existing bus. The previous section allows a device developer to choose the best transfer types for a new device and to estimate the percentage of the total bus that a new device will require. Devices should be "good citizens" on the bus to allow as much bandwidth as possible to be used by other devices.

This section describes some typical combinations of devices so a developer can understand what bus environments a new device can find itself in. The unused bus bandwidth is reported as the number of maximum sized (64 byte) bulk transactions that could additionally be moved during a frame. A developer can use other calculations are more specific to a device under investigation. But these estimates should give a feel for remaining unallocated real bandwidth available on USB under various loads.

per frame bytes	CDaudio+Mpeg2+8LS+ISDN (2B+D)						Bytes
	CDa	Mpeg	LS	ISDN	bit stuff	unalloc	Req'd
	190	**798**	**162**	**63**	**178**	-123	*1213*

Table 7 - Example Bus Bandwidth Requirements

The single device examples constructed above can be combined in several ways in example bus configurations. Three examples of bus configurations will be used to estimate the range of USB throughput that could be expected to be sustained:

- a light configuration consisting of a hub and a low speed keyboard. Where the hub could well be included in the keyboard as a compound device. This is essentially a vacant bus. Given the hot plug support of USB, this is a reasonable configuration since all devices can be detached at any time. This example configuration should represent the maximum available bandwidth that can be expected.
- a typical configuration consisting of the minimum configuration plus an additional hub (in a video monitor), CD audio output speakers, an ISDN telephony line and up to 7 additional low speed input devices (mice, joysticks, etc.). This example configuration represents a reasonable configuration that could well be encountered in an office setting when a new device is attached to a bus.

- a maximum configuration consisting of the typical configuration with the addition of an MPEG2 video camera. This example configuration represents the maximum bandwidth that could be carried over the bus. Device developers should not reasonably expect to exceed the bandwidth carried by this configuration.

Each configuration is presented in bar chart and tabular form. The bar chart gives an easily visible indication of the saturation of the bus, while the table allows more detailed analysis. The top of each bar chart shows the

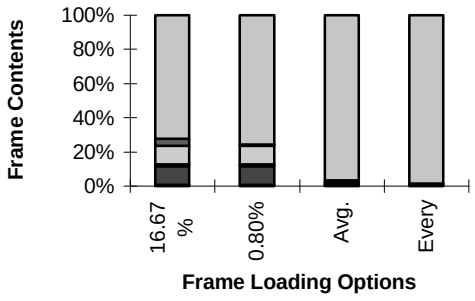

Figure 1.

remaining unallocated byte per frame.

Each example consists of four cases to take into account variations due to bit stuffing and frame to frame transaction differences. Each case illustrates different estimates of typical actual frames.

The first case assumes worst case bit stuffing (16.67%) and that every frame has every device transaction. This is a worst/worst case and will not normally be expected to occur in practice, but is possible.

The second case changes the bit stuffing assumption to 0.8%, more appropriate for random data. This is more typical of what will be seen in practice for bit stuffing. However, the transactions in each frame are still over estimated.

The next case again uses 0.8% bit stuffing, but a numerical average for the transaction load in each frame due to those transactions that vary from frame to frame, e.g. hubs won't typically be polled in every frame. This case is more reasonable of the average frame to frame load. However, it underestimates frame loading since transactions are continuously variable, only specific data payload sizes are possible for a given device.

The last case uses 0.8% bit stuffing and considers only those transactions that must be present in every frame. This is a best case for frame loading. It also under estimates transactions in a frame since most frames will have more transactions than this for a given configuration. Some frame at some time may have this load, but most will have more transactions.

Light Configuration

Figure 1 shows that even with worst case bit stuffing (labeled "16.67%") and assuming every frame has all maximum size transactions, over 70% of the bus is available for other device use. At the other extreme (labeled "Every"), over 95% of the frame time is available in more typical cases. The two middle barcharts show random data bit stuffing ("0.80%") and

Light Configuration

	16.67%	0.80%	Avg.	Every
frame mgt	14	14	13	13
LS Control	162	162	0	0
Hub Poll	16	16	2	0
LS Int	162	162	21	0
bitstuff	59	9	17	9
unalloc	1087	1137	1447	1478
64B Bulks	12	15	19	19
Bulk BW	768000	960000	1216000	1216000

Table 8. - Frame Loading, Light Configuration

average bit stuffing.

Table 8 contains the frame details corresponding to this configuration. The first five rows show the contribution to the frame time due to frame overhead, device data requirements (including transaction protocol overhead) and bit stuffing. The next row shows how many bytes per frame are unallocated and can be used for other devices.

The last two rows show the number of 64 byte sized bulk transactions that can fill the unallocated frame time and the resultant additional overall bandwidth carried in that case.

The average column for this example configuration is adjusted as follows:

- Frame management is reduced by 1 BTF assuming the on average the clock adjustment is no worse than -8 bits.
- Low speed control is assumed on average to not have any pending transfer requests.
- Hub poll and low speed interrupt times are divided by 8 on the assumption that a poll only occurs every 8 ms.
- Bit stuff is assumed to be 25% of worst case.

The "every frame" column shows that the hub poll and low speed interrupt transfers don't occur in every frame. Also bit stuffing is assumed to be the 0.8% minimum.

As Table 8 shows, this configuration has from 768KB/s to 1.2MB/s of deliverable unused bandwidth. For a reference point, Intel estimates that typical PC parallel port (Centronics, ECP, EPP, etc.) data rates range from 128KB/s to 1MB/s depending on particular implementations and operating system overhead. Typical PC serial port data rates are estimated by Intel to be limited to 115K bits/s or less.

Typical Configuration

Figure 2 shows that with the addition of CD quality audio and ISDN, frame loading is about 50% in worst/worst ("16.67%") case. However when more typical frame loading ("Avg.") is considered, frame loading is only about 20%.

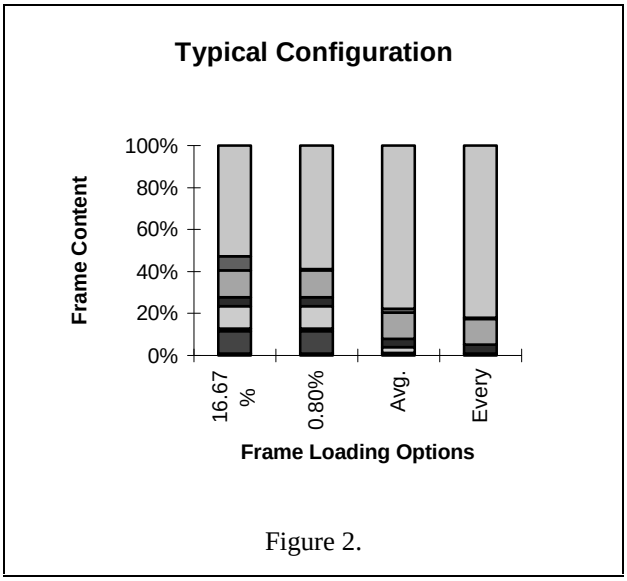

Figure 2.

Table 9 shows the details for Figure 2. New rows are added for ISDN and CD audio. The same adjustments are used for the average and every columns as in Table 8. Note that ISDN and CD audio don't vary significantly from frame to frame due their isochronous nature.

Even with the additional load of this example configuration, there is still 576KB/s to 1MB/s of bulk data transfer time available on the bus.

Typical Configuration

	16.67%	0.80%	Avg.	Every
frame mgt	14	14	13	13
LS Control	162	162	0	0
Hub Poll	16	16	4	0
LS Int	162	162	41	0
ISDN	63	63	63	63
CD-A	190	190	187	186
bitstuff	102	9	28	9
unalloc	791	884	1164	1229
64B Bulks	9	12	15	16
Bulk BW	576000	768000	960000	1024000

Table 9. - Frame Loading, Typical Configuration

In passing, it can be seen that ISDN only accounts for 63 bytes per frame or less than 5% of the bus. ISDN is a much higher speed device compared to other modem devices (e.g. 28.8Kb/s) and is difficult to support on existing PC serial ports, but is not a substantial load for USB.

Maximum Configuration

Figure 3 shows the frame loading with the addition of the 6Mb/s MPEG2 video camera. The camera imposed load is significant, over 50% of the bus. The design of this type of device needs to be considered carefully. A device that requires over about 30% of the bus bandwidth needs to be dealt with such that end users will only expect one to be in use at a time.

As will be seen in more detail below, when worst case bit stuffing and frame loading is assumed, this device cannot be reliably supported. However, if less bit stuffing or less frame loading is assumed, this

configuration is possible. In more typical frames, there is still 128KB/s to 256KB/s of unused bulk transfer

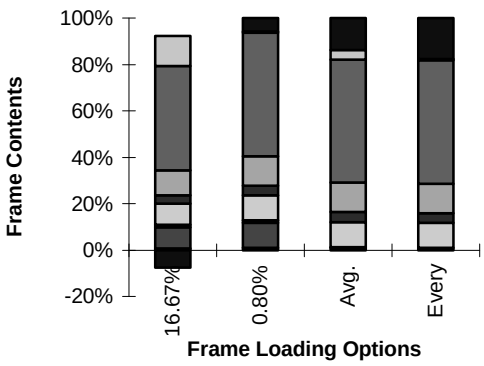

Figure 3.

time.

Table 10 shows that the worst/worst case configuration is 137 bytes too short to fit in a frame. However, not every frame will be fully loaded nor worst case bit stuffed. Bit stuffing at 16.67% requires 232 bytes. If more random bit stuffing is required, bit stuffing can be reduced to 9 bytes and even fully loaded frames are possible. The average and every frame cases show that there can be substantial bus time available for other devices.

Conclusion

The analysis presented here has estimated several example USB devices. Bus configurations consisting of several of these devices can be supported on USB. These estimates show that a broad range of practical bus configurations include a significant amount of available bandwidth that can be used for additional devices. Even in worst case configurations, significant bandwidth may be expected for non-guaranteed transfers such as those that may be needed for printers and scanners.

The throughput possibilities of USB should allow for a broad range of useful devices and bus configurations. End users should also be able to reliably make use of the devices and configurations they desire.

Spreadsheet formulas are available by email request from john_garney@ccm.jf.intel.com.

Acknowledgment

The author acknowledges the work of: Mike Dwyer for generating the original bus transmission timings and for performing the bit stuffing simulations, Jan Camps and Mike Bernstein for earlier presentation pictures, Shelagh Callahan for checking the telephony and isochronous estimates.

Heavy Configuration

	16.67%	0.80%	Avg.	Every
frame mgt	14	14	14	14
LS Control	162	162	0	0
Hub Poll	16	16	6	0
LS Int	162	162	162	162
ISDN	63	63	63	63
CD-A	190	190	190	190
MPEG	798	798	798	798
bitstuff	232	9	61	9
unalloc	-137	86	206	264
64B Bulks	0	2	3	4
Bulk BW	0	128000	192000	256000

Table 10. - Frame Loading, Heavy Configuration

References

[1] *Universal Serial Bus Specification, Version 1.0*. Available from http://www.teleport.com/~USB.