



White Paper: Interrupts and USB

In most embedded computer systems, there is a need for interrupts to handle events that require prompt handling by the operating system or application program. Although USB supports interrupt transfers, it is significantly different from the interrupts implemented on other bus architectures such as PCI or ISA.

USB interrupt transfers provide a guaranteed maximum latency communication pathway between the host and the USB device. An interrupt IN transfer is an interrupt transfer that is originated at the device and targeted at the host (direction is always referenced as viewed by the host). Interrupt IN transfers can be used by the device to alert the host of an important system event.

An interrupt transfer is not equivalent to an interrupt at one of the IRQ inputs of the host processor. As is the case with all transfers over USB the host must initiate the interrupt transfer. The device can make the interrupt transfer data available when a system event occurs, but the transfer does not start until the host requests the data. The way that interrupt transfer latency is guaranteed is that the host is obligated to poll for interrupt transfer data at a requested periodic interval. This interval is determined during enumeration and the host polls the device at this interval continuously after enumeration is complete.

The allowable range for interrupt transfer latency or host polling interval varies with USB bus speed. The following table shows the possible settings for each bus speed.

Bus Speed	Maximum Latency	bInterval units
High	125 usec – 4 sec	125 usec
Full	1 - 255 msec	1 msec
Low	10 - 255 msec	1 msec

From the table it can be seen that the smallest possible interrupt latency that can be achieved between a device and the host is 125 usec. The device requests interrupt latency by setting the bInterval field of the endpoint descriptor for the corresponding interrupt endpoint. The requested latency can be calculated by (bInterval) x (bInterval units).

The sum of all low- and full-speed interrupt and isochronous transfers is limited to consuming 90% of the USB bus bandwidth. For high speed, the limit is 80%. If a device is enumerated and its bInterval request puts the bus utilization over these limits, the host will refuse to configure the device.

Depending on many factors the host processor may not be able to transfer the interrupt data at the requested interval. OS design, driver design, application software, CPU speed, and bus bandwidth may all limit the host's ability to meet the obligation to poll for interrupt transfer data within the required interval.

A few large interrupt transfers are more efficient than a larger number of smaller interrupt transfers. Since most USB devices today are developed using a microcontroller, the microcontroller can be used to queue up the data and make it available to the host in larger transfers, thereby decreasing the number of transfers and increasing the size of each transfer and increasing efficiency.