Keywords: USB, MAX3420E, MAX3421E, Terminology

Apr 10, 2006

APPLICATION NOTE 3803

Practical USB Terminology

Abstract: If you are new to USB, it will be helpful to understand some USB terminology before delving into Maxim's USB application notes. This application note explains basic USB terminology, with specific relevance to the MAX3420E USB controller and the MAX3421E USB peripheral/host controller.

Host

USB is a "master-slave" bus with exactly one master and multiple slaves. The slaves are called peripherals or in USB nomenclature, functions. The master is called a host. Only the host can initiate USB transfers; the peripherals always respond, never initiate. A PC is the most common host. Hosts connect to *downstream* devices using USB-A connectors. An embedded host does not involve a PC, but instead uses a microcontroller to implement a special-purpose host, perhaps to talk to just one class of USB devices.

The MAX3421E connects to a controller to implement an embedded host.

Function

A function is an USB device, also called an USB peripheral. USB peripherals are "downstream" from the host, and use the USB B-connector.

- The MAX3420E is a peripheral controller.
- The MAX3421E can operate either as an USB host or peripheral controller.

Speed

The USB 2.0 Specification defines three speeds:

1. Low speed is 1.5Mbps, used mostly by keyboards and mice.
2. Full speed is 12Mbps.
3. High speed is 480Mbps.

There is some confusion in the marketplace regarding the "USB 2.0 Compatible" claim. The confusion stems from the progression of USB specification versions. First there was USB Specification 1.0, and soon thereafter version 1.1 which clarified the 1.0 specification. Version 1.X contained two USB bus speeds, low-speed and full-speed. Version 2.0 added high-speed, and *replaced the 1.1 specification in its entirety*. Therefore, if you make a full-speed device (12Mbps), you can correctly claim to be compatible with USB Specification version 2.0, even though many people associate USB 2.0 only with high-speed (480Mbps) operation.

- The MAX3420E operates as a full-speed peripheral device.
- The MAX3421E operates as a full-speed peripheral device or a full-/low-speed host.

IN-OUT Direction

An USB system is host-centric. Consequently, USB nomenclature assumes that you are talking about the host. Therefore, an IN transfer is from the host's point of view, that is a transfer from the peripheral to the host. Likewise, an OUT transfer travels from the host to the peripheral.

It may seem somewhat strange when writing MAX3420E code to load an "INFIFO" with data to be sent from the chip. However, if you remember that the host is the master, it makes sense.

Endpoint

An endpoint is the source or sink of data inside an USB peripheral device; it is an addressable FIFO. An USB peripheral has a unique address and can contain up to sixteen endpoints. The host sends data to a specific endpoint (FIFO) by including a device address and endpoint number in every data transfer.

Each endpoint has an address from 0 to 15 and a direction. Therefore, Endpoint 2-IN is a completely different entity than Endpoint 2-OUT. Every device has a default CONTROL Endpoint 0 which is bidirectional, so there is no separate Endpoint 0-IN and Endpoint 0-OUT.

Endpoints are arbitrarily numbered inside the peripheral device. The peripheral reports its endpoint numbers and characteristics to the host during enumeration.

The MAX3420E provides the following endpoints:

- EP0, the default CONTROL endpoint
- EP1-OUT
- EP2-IN
- EP3-IN

The MAX3421E provides the same endpoints as the MAX3420E when operating as a peripheral. When operating as a host, however, the MAX3421E uses a SNDFIFO and RCVFIFO to transfer data to any peripheral address and endpoint. The MAX3421E firmware loads registers with the function address and desired endpoint number, then sends the packet(s).

Hub

Hubs expand the number of devices to which an USB host can talk. A PC has a root hub built into its USB controller, which does some USB low-level work like detecting devices plugging in and out. Hubs talk upstream (to the PC) at full speed or high speed, and downstream to peripherals at all three speeds. The PC's root hub can supply 500mA of 5V power per A-connector. A bus-powered external hub (no wall wart) can supply 100mA per port. USB allocates 100mA for external hub circuitry, so the most downstream ports that a bus-powered hub can have are four. If the hub has its own power supply (self-powered), each downstream port can supply up to 500mA.

- MAX3420E firmware is unaware of how many hub levels to which it may be connected.
- The MAX3421E provides a special mode to talk to a low-speed peripheral that is connected to a hub.

Enumeration

When you plug in an USB device, the host gets a connection notice and proceeds to determine what just plugged in. The host requests a series of descriptors (table data) and the device returns them, all over the default CONTROL Endpoint 0. If the host is satisfied with what it receives from the device, it configures the device for operation. If the host is not satisfied with the device's data (for example, if any of the descriptor data was inconsistent or out of spec), it ignores the device. You see a small popup window saying that there is a problem with this USB device.

Chapter 9 (More About Enumeration)

Chapter 9 of the USB Specification defines all the requests that a host sends to a peripheral during enumeration, and the data formats for the peripheral responses. If you visit the official USB website, you will find a software tool called USBCV (USB Command Verifier) that includes a section called "Chapter 9 Tests." These tests verify that your enumeration code is correct. USBCV is the same test used by certified USB testing labs. so If you pass this part of the USB certification in your lab, you will pass it in the test lab.

Maxim provides an application note for the MAX3420E with example enumeration code that passes the USBCV tests.

SIE

SIE indicates the Serial Interface Engine, the "guts" inside any USB controller. The SIE performs low-level details like bit stuffing, CRC generation and checking, and error reporting. The main role of an SIE is to convert the low-level signaling into bytes for use by the attached controller. Some SIEs are smarter than others. The more low-level detail it handles, the simpler the controlling firmware can be. Some SIEs, for example, only report the results of endpoint data toggles (see the following section), and leave it to the firmware to determine how to handle them. The MAX3420/MAX3421 handle data toggles inside the SIE.

Data Toggles

USB packets are prefaced with a PID, or Packet ID. Data transfers utilize two PIDs: DATA0 and DATA1. Both the host and peripheral contain data toggle bits, one per endpoint. The toggle bits determine which of these data PIDs should be used for the data transfer. When the peripheral comes out of reset, both sides, host and peripheral, reset their internal data toggle bits to zero. The first data packet is, therefore, sent using the DATA0 PID.

When a data packet transfers without errors (signaled when the sending side receives the ACK PID), both sides complement their data toggle values. The second data packet for the endpoint is then sent using the DATA1 PID. As successful transfers occur, the data packets use alternating (or toggling) DATA0 and DATA1 PIDs. USB uses this mechanism as part of its error correction.

- The MAX3420E automatically maintains the data toggles. Firmware only needs to get involved when the device is reconfigured or the host sends the Clear_Feature (ENDPOINT HALT) request. The MAX3420E contains register bits to clear each endpoint's toggle bits.
- When operating as a host, the MAX3421E maintains the data toggle value once the toggle value has been correctly set for the endpoint. The firmware typically saves the toggle value after completing a transfer to a particular endpoint. It then initializes the toggle prior to transferring data to the same endpoint. During multipacket transfers to the same endpoint, the SIE maintains the data toggle values.

CONTROL, BULK, INTERRUPT, ISOCHRONOUS Transfers

USB has the above four transfer types. During enumeration a peripheral tells the host which transfer type each of its endpoints supports.

Only hosts send CONTROL transfers, which comprise two or three stages. A SETUP packet sends the specific host request using an 8-byte data packet. An optional data packet moves data such as descriptor tables. Finally, a handshake (status) packet terminates the CONTROL transfer. CONTROL transfers, as "mission critical" transfers, have high bus priority and the most comprehensive error checking. Every USB peripheral requires a default CONTROL endpoint zero.

BULK transfers move data using flow control and error checking. BULK transfers are asynchronous, meaning that their scheduled transfer times are not regular or guaranteed. The host schedules BULK transfers as its lowest priority. This does not mean that BULK transfers are slow; if the bus is lightly loaded, the BULK transfers are scheduled to use all available bandwidth.

INTERRUPT transfers are, practically speaking, indistinguishable from BULK transfers. The only difference between them is that an INTERRUPT endpoint contains a polling interval value, which tells the host how often to "ping" the endpoint. Therefore the only practical difference between BULK and INTERRUPT transfers is how often the host schedules the transfers.

ISOCHRONOUS (ISO) transfers are for streaming data such as audio or video, where the data must arrive in time to avoid breaks in the audio or video. When a device enumerates it tells the host how much bandwidth its ISO endpoints require. If the bandwidth is available, the host guarantees that every 1ms USB frame will contain an ISO data packet to or from the device. ISO does not use handshakes (ACK/NAK) or bus retries. ISOCHRONOUS transfers are defined only for full-speed and high-speed transfers.

- The MAX3420E (or MAX3421E in peripheral mode) has the default CONTROL Endpoint 0, and three other endpoints that can be used as BULK or INTERRUPT Endpoints.
- The MAX3421E (host) generates transfers to all four endpoint types.

Bus Reset

The host resets an USB peripheral by asserting a bus reset. Full- and low-speed USB uses mostly differential signaling on its D+ and D- wires. The exception to differential signaling are the bus-reset and end-of-packet signals which use a single-ended-zero where D+ and D- are both low.

- The MAX3420E *detects* a bus reset and asserts an interrupt.
- The MAX3421E (host) *generates* a bus reset when the attached microcontroller sets a register bit, and then waits for a completion interrupt.

USB Class Driver

Windows® has built-in driver support for various USB device classes. If you write firmware that conforms to one of these class specifications, you do not need to supply a custom driver along with your USB product. (Nobody really wants to write Windows drivers.) Two common USB class drivers are HID (Human Interface Device) and mass storage (disk drives, CDROMS, memory sticks).

Maxim provides an application note showing how to write code to be compliant with the HID class.

Related Parts

MAX3420E:	QuickView	-- Full (PDF) Data Sheet	-- Free Samples
MAX3421E:	QuickView		-- Free Samples

Automatic Updates
Would you like to be automatically notified when new application notes are published in your areas of interest? Sign up for EE-Mail™.

Application note 3803: www.maxim-ic.com/an3803
More Information
For technical support: www.maxim-ic.com/support
For samples: www.maxim-ic.com/samples
Other questions and comments: www.maxim-ic.com/contact

AN3803, AN 3803, APP3803, Appnote3803, Appnote 3803