18

Signals and Encoding

You can design and program a USB peripheral without knowing all of the details about how the data is encoded on the bus. But understanding something about these topics can help in understanding the capabilities and limits of your devices.

This chapter presents the essentials of the USB's encoding and data formats. The USB specification has the details.

Bus States

The USB specification defines bus states that correspond either to signal voltages on the bus or conditions that these voltages signify. Different cable segments on a bus may be in different states at the same time. For example, in response to a request from the host, a hub might place one of its downstream ports in the Reset state while its other ports are in the Idle state. Low/full speed and high speed each have different defined bus states, though with many similarities.

Low-speed and Full-speed Bus States

Low and full speed support the same bus states, though some are defined differently depending on the speed of the cable segment. A low-speed segment is a segment between a low-speed device and its nearest hub. A full-speed segment is any other segment that carries data at low- or full-speed bit rates.

Differential 0 and Differential 1

When transferring data, the two states on the bus are Differential 0 and Differential 1. A Differential 0 exists when D+ is a logic low and D- is a logic high. A Differential 1 exists when D+ is a logic high and D- is a logic low.Chapter 19 has details about the voltages that define logic low and high.

The Differential 0s and 1s don't translate directly into voltage levels, but instead indicate either a change in logic level, no change in logic level, or a bit stuff, as explained later in this chapter.

Single-Ended Zero

The Single-Ended-Zero (SE0) state occurs when both D+ and D- are logic low. The bus uses the SingleEnded-Zero state when entering the End-of-Packet, Disconnect, and Reset states.

Single-Ended One

The complement of the Single-Ended Zero is the Single-Ended One (SE1). This state occurs when both D+ and D- are logic high. This is an invalid bus state and should never occur.

Data J and Data K

In addition to the Differential 1 and 0 states, which are defined by voltages on the lines, USB also defines two Data bus states, J and K. These are defined by whether the bus state is Differential 1 or 0 and whether the cable segment is low or full speed:

Bus State	Data State	
	Low Speed	Full Speed
Differential 0	J	K
Differential 1	K	J

Defining the J and K states in this way makes it possible to use one terminology to describe an event or logic state even though the voltages on low- and full-speed lines differ. For example, a Start-of-Packet state exists when the bus changes from Idle to the K state. On a full-speed segment, the state occurs when D- becomes more positive than D+, while on a low-speed segment, the state occurs when D+ becomes more positive than D-.

Idle

In the Idle state, no drivers are active. On a full-speed segment, D+ is more positive than D-, while on a low-speed segment, D- is more positive than D+. Shortly after device attachment, a hub determines whether a device is low or full speed by checking the voltages on the Idle bus at the device's port.

Resume

When a device is in the Suspend state, the Data K state at the device's port signifies a resume from Suspend.

Start-of-Packet

The Start-of-Packet (SOP) bus state exists when the lines change from the Idle state to the K data state. Every transmitted low- or full-speed packet begins with a Start of Packet.

End-of-Packet

The End-of-Packet (EOP) state exists when a receiver has been in the Single-Ended-Zero state for at least one bit time, followed by a Data J state for at least one bit time. A receiver may optionally define a shorter minimum time for the Data J state. At the driver, the Single-Ended Zero is approximately two bit widths. Every transmitted low- or full-speed packet ends with an End of Packet.

Disconnect

A downstream port is in the Disconnect state when a Single-Ended Zero has lasted for at least 2.5 microseconds.

Connect

A downstream port enters the Connect state when the bus has been in the Idle state for at least 2.5 microseconds and no more than 2.0 milliseconds.

Reset

When a Single-Ended Zero has lasted for 10 milliseconds, the device must be in the Reset state. A device may enter the Reset state after the Single-Ended Zero has lasted for as little as 2.5 microseconds. A full-speed device that is capable of high-speed communications performs the high-speed handshake during the Reset state.

On exiting the Reset state, a device must be operating at its correct speed and must respond to communications directed to the default address (00h).

High-speed Bus States

Many of the high-speed bus states are similar to those for low and full speed. A few are unique to high speed, and some low- and full-speed bus states have no equivalent at high speed.

High-speed Differential 0 and Differential 1

The two bus states that exist when transferring high-speed data are High-speed Differential 0 and High-speed Differential 1. As with low and full speeds, a High-speed Differential 0 exists when D+ is a logic low and D- is a logic high, and a High-speed Differential 1 exists when D+ is a logic high and D- is a logic low. The voltage requirements differ at high speed, however, and high speed has additional requirements for AC differential levels.

High-speed Data J and Data K

The definitions for High-speed Data J and Data K states are identical to those for full-speed J and K:

Bus State	Data State, High Speed
Differential 0	K
Differential 1	J

Chirp J and Chirp K

The Chirp J and Chirp K bus states are present only during the high-speed detection handshake. The handshake occurs when a 2.0 hub has placed a downstream bus segment in the Reset state. Chirp J and Chirp K are defined as DC differential voltages. In a Chirp J, D+ is more positive than D-, and in a Chirp K, D- is more positive than D+.

A high-speed device must use full speed on attaching to the bus. The high-speed detection handshake enables a high-speed device to tell a 2.0 hub that the device supports high speed and to transition to high-speed communications.

As Chapter 4 explained, shortly after detecting device attachment, a device's hub places a device's port and bus segment in the Reset state. When a high-speed-capable device detects the Reset, the device sends a Chirp K to the hub for 1 to 7 milliseconds. A 2.0 hub that is communicating upstream at high speed detects the Chirp K and in response, sends an alternating sequence of Chirp Ks and Js. The sequence continues until shortly before the Reset state ends. At the end of Reset, the hub places the port in the High-speed Enabled state.

On detecting the Chirp K and Chirp J sequence, the device disconnects its full-speed pull-up, enables its high-speed terminations, and enters the high-speed Default state.

A 1.x hub ignores the device's Chirp K. The device doesn't see the answering sequence and knows that communications must take place at full speed.

High-speed Squelch

The High-speed Squelch state indicates an invalid signal. High-speed receivers must include circuits that detect the Squelch state, indicated by a differential bus voltage of 100 millivolts or less.

High-speed Idle

In the High-speed Idle state, no high-speed drivers are active and the low/full-speed drivers assert Single-Ended Zeroes. Both D+ and D- are between -10 and +10 millivolts.

Start of High-speed Packet

A Start-of-High-speed-Packet (HSSOP) exists when a segment changes from the High-speed Idle state to the High-speed Data K state. Every high-speed packet begins with a Start of High-speed Packet.

End of High-speed Packet

An End-of-High-speed-Packet (HSEOP) exists when the bus changes from the High-speed Data K or Data J state to the High-speed Idle state. Every high-speed packet ends with an End of High-speed Packet.

High-speed Disconnect

Removing a high-speed device from the bus also removes the high-speed line terminations at the device. The removal of the terminations causes the differential voltage at the hub to double. A differential voltage of 625 millivolts or more on the data lines indicates the High-speed Disconnect state. A 2.0 hub contains circuits that detect this voltage.

Data Encoding

All data on the bus is encoded. The encoding format, called *Non-Return to Zero Inverted (NRZI) with bit stuffing,* ensures that the receiver remains synchronized with the transmitter without the overhead of sending a separate clock signal or Start and Stop bits with each byte.

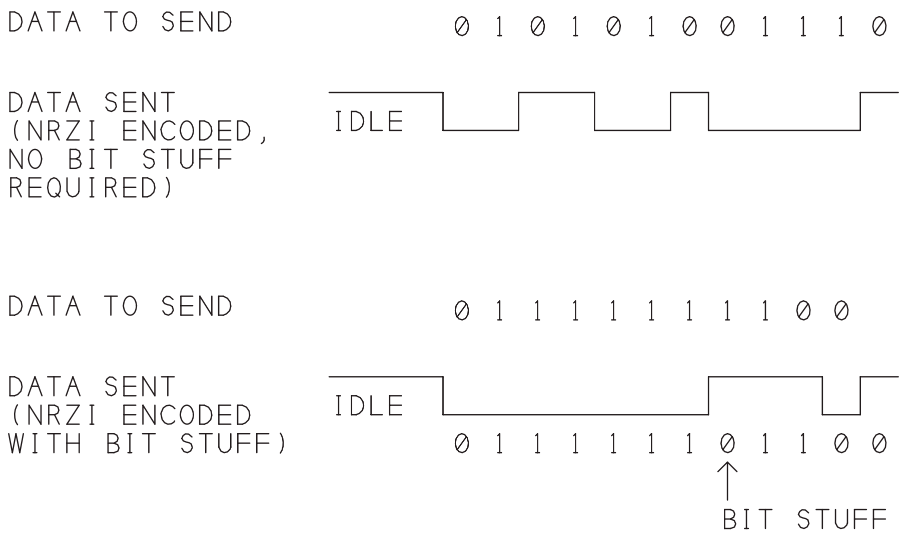

Figure 18-1: In NRZI encoding, a 0 causes a change and a 1 causes no change. Bit stuffing adds a 0 after six consecutive 1s.

If you use an oscilloscope or logic analyzer to view USB data on the bus, you'll find that unlike some other interfaces, reading the bits isn't as easy as matching voltage levels to logic levels.

Instead of defining logic 0s and 1s as voltages, NRZI encoding defines logic 0 as a voltage change, and logic 1 as a voltage that remains the same. Figure 18-1 shows an example. Each logic 0 results in a change from the previous state. Each logic 1 results in no change in the voltages. The bits transmit least-significant-bit (LSB) first.

Fortunately, the available USB hardware does all of the encoding and decoding automatically, so device developers and programmers don't have to worry about it. The encoded data makes it difficult to interpret the data on an oscilloscope or logic analyzer, but as Chapter 17 showed, the solution is to use a protocol analyzer that decodes the data for you.

Staying Synchronized

When two devices exchange data, the receiving device needs a way to know when each bit is available to be read. With the RS-232 interface, the transmitter and receiver each have their own clock reference, and both must agree on a bit rate for exchanging data. Each transmitted word begins with a transition from the Idle state to a Start bit. The receiver synchronizes to this transition and then uses timing circuits and the agreed-on bit rate to read each bit in the middle of each bit time. The Stop bit returns the link to the Idle state so the next Start bit can be detected. If the transmitter's and receiver's clocks differ by up to a few percent, the receiver will still be able to read ten or eleven bits before a new Start bit resynchronizes the clocks. But adding a Start and Stop bit to each data byte adds 25 percent overhead. A 9600-bps link with 8 data bits and one Start and Stop bit transmits only 7680 data bits (960 bytes) per second.

Another approach used by SPI, I^2C, and Microwire interfaces is to send a clock signal along with the data. The receiver detects the bits either on detecting a rising or falling edge or a high or low logic level, depending on the protocol. Sending a clock requires an extra signal line, however, and a noise glitch on the clock line can cause misread data.

The NRZI encoding used in USB communications requires no Start and Stop bits or clock line. Instead, USB uses two other techniques to remain synchronized: bit stuffing and SYNC fields. Each adds some overhead to each transaction, but the amount is minimal with large packets.

Bit Stuffing

Bit stuffing is required because the receiver synchronizes on transitions. If the data is all 0s, there are plenty of transitions. But if the data contains a long string of 1s, the lack of transitions could cause the receiver to get out of sync.

If data has six consecutive 1s, the transmitter stuffs, or inserts, a 0 (represented by a transition) after the sixth 1. This ensures at least one transition for every seven bit widths. The receiver detects and discards any bit that follows six consecutive 1s.

Bit stuffing can increase the number of transmitted bits by up to 17 percent. In practice the average is much less. The bit-stuffing overhead for random data is just 0.8 percent, or one stuff bit per 125 data bits.

SYNC Field

Bit stuffing alone isn't enough to ensure that the transmitting and receiving clocks in a transfer are synchronized. Because devices and the host don't share a clock, the receiving device has no way of knowing exactly when a transmitting device will send a transition that marks the beginning of a new packet. A single transition isn't enough to ensure that the receiver will remain synchronized for the duration of a packet.

To keep things synchronized, each packet begins with a SYNC field to enable the receiving device to align, or synchronize, its clock to the transmitted data. For low and full speeds, the SYNC pattern is eight bits: KJKJKJKK. The transition from Idle to the first K serves as a sort of Start bit that indicates the arrival of a new packet. There's one SYNC field per packet, rather than a Start bit for each byte.

For high speed, the SYNC pattern is 32 bits: fifteen KJ repetitions, followed by KK. A high-speed hub repeating a packet can drop up to four bits from the beginning of the sync field, so a SYNC field repeated by the fifth external hub series can be as short as 12 bits.

The alternating Ks and Js provide the transitions for synchronizing, and the final two Ks mark the end of the field. By the end of the SYNC pattern, the receiving device can determine precisely when each of the remaining bits in the packet will arrive. The price to pay for synchronizing is the addition of 8 to 32 bit times to each packet. Large packets are thus much more efficient than smaller ones.

End of Packet

An End-of-Packet signal returns the bus to the Idle state in preparation for the next SYNC field. The End-of-Packet signal is different for low/full and high speed.

The low- or full-speed End of Packet is a Single-Ended-Zero that lasts for two bit widths.

At high speed, the signal is more complicated. High-speed receivers treat any bit-stuff error as an End of Packet, so an End of High-speed Packet must cause a bit-stuff error.

For all high-speed packets except Start-of-Frame packets, the End of High-speed Packet is an encoded byte of 01111111, without bit stuffing. If the preceding bit was a J, the End of High-speed Packet is KKKKKKKK. The initial 0 causes the first bit to be a change of state from J to K, and the following 1s mean that the rest of the bits don't change. If the preceding bit was a K, the End of High-speed Packet is JJJJJJJJ. The initial 0 causes the first bit to be a change of state from K to J, and the following 1s mean that the rest of the bits don't change. In either case, the sequence of seven 1s causes a bit stuff error.

In high-speed Start-of-Frame packets, the End of High-speed Packet is 40 bits. This allows a hub time to detect the doubled differential voltage that indicates that a device has been removed from the bus. The encoded byte begins with a zero, followed by 39 ones, which results in an End of High-speed Packet consisting of 40 Js or 40 Ks. As with low and full speeds, this sequence results in a bit-stuff error that the receiver treats as an End of Packet.

Timing Accuracy

A tradeoff of speed is more stringent timing requirements. USB's high speed has the most critical timing, followed by full speed and then low speed, which is quite tolerant of timing variations.

Devices typically derive their timing from a crystal. Many factors can affect a crystal's frequency, including initial accuracy, capacitive loading, aging of the crystal, supply voltage, and temperature. Crystal accuracy is typically specified as parts per million (ppm), which is the maximum number of cycles the crystal may vary in the time required for 1 million cycles at the rated frequency.

High speed's bit rate of 480 Megabits/sec. can vary no more than 0.05 percent, or 500 ppm. Full speed's bit rate of 12 Megabits/sec. can vary no more than 0.25 percent, or 2500 ppm. Low speed's bit rate of 1.5 Megabit/sec. can vary up to 1.5%, or 15,000 ppm. The greater tolerance for low speed means that low-speed devices can use inexpensive ceramic resonators instead of quartz crystals.

The data rate at a host or 2.0 hub must be within 0.05%, or 500 ppm, of the specified rate at all speeds. The frame intervals must be accurate as well, at 1 millisecond ±500 nanoseconds per frame or 125.0 ±62.5 microseconds per microframe. To maintain this accuracy, hubs must be able to adjust their frame intervals to match the host's. Each hub has its own timing source and synchronizes its transmissions to the host's Start-of-Frame signals in each frame or microframe.

The USB specification also defines limits for data jitter, or small variations in the timing of the individual bit transitions. The limits allow small differences in the rise and fall times of the drivers as well as clock jitter and other random noise.

Packet Format

As Chapter 2 explained, all USB data travels in packets, which are blocks of information with a defined format. The packets in turn contain fields, with each field type holding a particular type of information.

Fields

Table 18-1 lists the fields that packets contain and their purposes.

SYNC

Each packet begins with an 8-bit SYNC field, as described earlier. The SYNC Field serves as the Start-of-Packet delimiter.

Table 18-1: All USB traffic is in packets. Packets are made up of fields. The field type determines its contents.

Name	SIze (bits)	Packet Types	Purpose
SYNC	8	all	Start-of-packet and synchronization
PID	8	all	Identify the packet type
Address	7	IN, OUT, Setup	Identify the function address
Endpoint	4	IN, OUT, Setup	Identify the endpoint
Frame Number	11	SOF	Identify the frame
Data	0 to 8192 (1024 bytes) for 2.0 hardware; 0 to 8184 (1023 bytes) for 1.x hardware	Data0, Data1	Data
CRC	5 or 16	IN, OUT, Setup, Data0, Data1	Detect errors

Packet Identifier

The packet identifier field (PID) is 8 bits. Bits 0 through 3 identify the type of packet and bits 4 through 7 are the one's complement of these bits, for use in error checking.

There are 16 defined PID codes for token, data, handshake and special packets. Chapter 2 introduced these codes. The lower two bits identify the PID type, and the upper two bits identify the specific PID.

Address

The address field is seven bits that identify the device the host is communicating with.

Endpoint

The endpoint field is four bits that identify an endpoint number within a device.

Frame Number

The frame-number field is eleven bits that identify the specific frame. The host sends this field in the Start-of-Frame packet that begins each frame or microframe. After 07FFh, the number rolls over to zero. A full-speed host maintains an 11-bit counter that increments once per frame. A high-speed host maintains a 14-bit counter that increments once per microframe. Only bits 3–13 of the microframe counter transmit in the frame number field, so the frame number increments once per frame, with eight microframes in sequence having the same frame number.

Data

The data field may range from 0 to 1024 bytes, depending on the transfer type, the bus's speed, and the amount of data in the transaction.

CRC

The CRC field is 5 bits for address and endpoint fields and 16 bits for data fields. The bits are used in error-checking. The transmitting hardware normally inserts the CRC bits and the receiving hardware does the required calculations; there's no need for program code to do it.

Inter-packet Delay

USB carries data from multiple sources, in both directions, on one pair of wires. Data can travel in just one direction at a time. To ensure that the previous transmitting device has had time to switch off its driver, the bus requires a brief delay between the end of one packet and the beginning of the next packet in a transaction. This delay time is limited, however, and devices must switch directions quickly.

The USB specification defines the delays differently for low/full and high speed. The delays are handled by the hardware and require no support in code.

Test Modes

For use in compliance testing, the USB 2.0 specification adds five new test modes that all host controllers, hubs, and high-speed-capable devices must support.

Entering and Exiting Test Modes

An upstream-facing port enters a test mode in response to a Set_Feature request with TEST_MODE in the wValue field. A downstream-facing port enters a test mode in response to the hub-class request Set_Port_Feature with PORT_TEST in the wValue field. In both cases, the wIndex field contains the port number and the test number. All downstream ports on a hub with a port to be tested must be in the suspended, disabled, or disconnected state.

An upstream-facing port exits the test mode when the device powers down and back up. A downstream-facing port exits the test mode when the hub is reset.

The Modes

These are the five test modes:

Test_SEO_NAK

Value. 01h.

Action. The transceiver enters and remains in high-speed receive mode. Upstream-facing ports respond to IN token packets with NAK.

Purpose. Test output impedance, low-level output voltage, and loading characteristics. Test device squelch-level circuits. Provide a stimulus-response test for basic functional testing.

Test_J

Value. 02h.

Action. The transceiver enters and remains in the High-speed Data J state.

Purpose. Test the high output drive level on D+.

Test_K

Value. 03h.

Action. The transceiver enters and remains in the High-speed Data K state.

Purpose. Test the high output drive level on D-.

Test_Packet

Value. 04h.

Action. Repetitively transmit the test packet defined by the USB specification.

Purpose. Test rise and fall times, eye pattern, jitter, and other dynamic waveform specifications.

Test_Force_Enable

Value. 05h.

Action. Enable downstream-facing hub ports in high-speed mode. Packets arriving at the upstream-facing port are repeated at the port being tested. The disconnect-detect bit can be polled while varying the loading on the port.

Purpose. Measure the disconnect-detection threshold.

Other Values

Test-mode values 06h through 3Fh are reserved for future standard tests. Value C0h through FFh are available for vendor-defined tests. All other values are reserved.