Table 6-7. USB Electrical, Mechanical, and Environmental Compliance Standards (Continued)

Test Description	Test Procedure	Performance Requirement
Flammability	UL 94 V-0 This procedure is to ensure thermoplastic resin compliance to UL flammability standards.	The manufacturer will require its thermoplastic resin vendor to supply a detailed C of C with each resin shipment. The C of C shall clearly show the resin's UL listing number, lot number, date code, etc.
Flammability	UL 94 V-0 This procedure is to ensure thermoplastic resin compliance to UL flammability standards.	The manufacturer will require its thermoplastic resin vendor to supply a detailed C of C with each resin shipment. The C of C shall clearly show the resin's UL listing number, lot number, date code, etc.
Cable Impedance (Only required for high-/full-speed)	The object of this test is to insure the signal conductors have the proper impedance. 1. Connect the Time Domain Reflectometer (TDR) outputs to the impedance/delay/skew test fixture (Note 1). Use separate 50 Ω cables for the plus (or true) and minus (or complement) outputs. Set the TDR head to differential TDR mode. 2. Connect the Series "A" plug of the cable to be tested to the text fixture, leaving the other end open-circuited. 3. Define a waveform composed of the difference between the true and complement waveforms, to allow measurement of differential impedance. 4. Measure the minimum and maximum impedances found between the connector and the open circuited far end of the cable.	Impedance must be in the range specified in Table 7-9 (Z_O).

Table 6-7. USB Electrical, Mechanical, and Environmental Compliance Standards (Continued)

Test Description	Test Procedure	Performance Requirement
Signal Pair Attenuation (Only required for high-/full-speed)	The object of this test is to insure that adequate signal strength is presented to the receiver to maintain a low error rate. 1. Connect the Network Analyzer output port (port 1) to the input connector on the attenuation test fixture (Note 2). 2. Connect the Series "A" plug of the cable to be tested to the test fixture, leaving the other end open-circuited. 3. Calibrate the network analyzer and fixture using the appropriate calibration standards over the desired frequency range. 4. Follow the method listed in Hewlett Packard Application Note 380-2 to measure the open-ended response of the cable. 5. Short circuit the Series "B" end (or bare leads end, if a captive cable) and measure the short-circuit response. 6. Using the software in H-P App. Note 380-2 or equivalent, calculate the cable attenuation accounting for resonance effects in the cable as needed.	Refer to Section 7.1.17 for frequency range and allowable attenuation.

Table 6-7. USB Electrical, Mechanical, and Environmental Compliance Standards (Continued)

Test Description	Test Procedure	Performance Requirement
Propagation Delay	The purpose of the test is to verify the end to end propagation of the cable. 1. Connect one output of the TDR sampling head to the D+ and D- inputs of the impedance/delay/skew test fixture (Note 1). Use one 50 Ω cable for each signal and set the TDR head to differential TDR mode. 2. Connect the cable to be tested to the test fixture. If detachable, plug both connectors in to the matching fixture connectors. If captive, plug the series "A" plug into the matching fixture connector and solder the stripped leads on the other end to the test fixture. 3. Measure the propagation delay of the test fixture by connecting a short piece of wire across the fixture from input to output and recording the delay. 4. Remove the short piece of wire and remeasure the propagation delay. Subtract from it the delay of the test fixture measured in the previous step.	High-/full-speed. See Section 7.1.1.1, Section 7.1.4, Section 7.1.16, and Table 7-9 (TFSCBL). Low-speed. See Section 7.1.1.2, Section 7.1.16, and Table 7-9 (TLSCBL).

Table 6-7. USB Electrical, Mechanical, and Environmental Compliance Standards (Continued)

Test Description	Test Procedure	Performance Requirement
Propagation Delay Skew	This test insures that the signal on both the D+ and D- lines arrive at the receiver at the same time. 1. Connect the TDR to the fixture with test sample cable, as in the previous section. 2. Measure the difference in delay for the two conductors in the test cable. Use the TDR cursors to find the open-circuited end of each conductor (where the impedance goes infinite) and subtract the time difference between the two values.	Propagation skew must meet the requirements as listed in Section 7.1.3.
Capacitive Load Only required for low-speed	The purpose of this test is to insure the distributed inter-wire capacitance is less than the lumped capacitance specified by the low-speed transmit driver. 1. Connect the one lead of the Impedance Analyzer to the D+ pin on the impedance/delay/skew fixture (Note 1) and the other lead to the D- pin. 2. Connect the series "A" plug to the fixture, with the series "B" end leads open-circuited. 3. Set the Impedance Analyzer to a frequency of 100 kHz, to measure the capacitance.	See Section 7.1.1.2 and Table 7-7 (CLINUA).

Note1: Impedance, propagation delay, and skew test fixture
This fixture will be used with the TDR for measuring the time domain performance of the cable under test. The fixture impedance should be matched to the equipment, typically 50 Ω. Coaxial connectors should be provided on the fixture for connection from the TDR.

Note 2: Attenuation text fixture
This fixture provides a means of connection from the network analyzer to the Series "A" plug. Since USB signals are differential in nature and operate over balanced cable, a transformer or balun (North Hills NH13734 or equivalent) is ideally used. The transformer converts the unbalanced (also known as single-ended) signal from the signal generator which is typically a 50 Ω output to the balanced (also known as differential) and likely different impedance loaded presented by the cable. A second transformer or balun should be used on the other end of the cable under test to convert the signal back to unbalanced form of the correct impedance to match the network analyzer.

6.7.1 Applicable Documents

American National Standard/Electronic Industries Association

ANSI/EIA-364-C (12/94)	Electrical Connector/Socket Test Procedures Including Environmental Classifications

American Standard Test Materials

ASTM-D-4565	Physical and Environmental Performance Properties of Insulation and Jacket for Telecommunication Wire and Cable, Test Standard Method
ASTM-D-4566	Electrical Performance Properties of Insulation and Jacket for Telecommunication Wire and Cable, Test Standard Method

Underwriters' Laboratory, Inc.

UL STD-94	Test for Flammability of Plastic materials for Parts in Devices and Appliances
UL Subject-444	Communication Cables

6.8 USB Grounding

The shield must be terminated to the connector plug for completed assemblies. The shield and chassis are bonded together. The user selected grounding scheme for USB devices, and cables must be consistent with accepted industry practices and regulatory agency standards for safety and EMI/ESD/RFI.

6.9 PCB Reference Drawings

The drawings in Figure 6-12, Figure 6-13, and Figure 6-14 describe typical receptacle PCB interfaces. These drawings are included for informational purposes only.

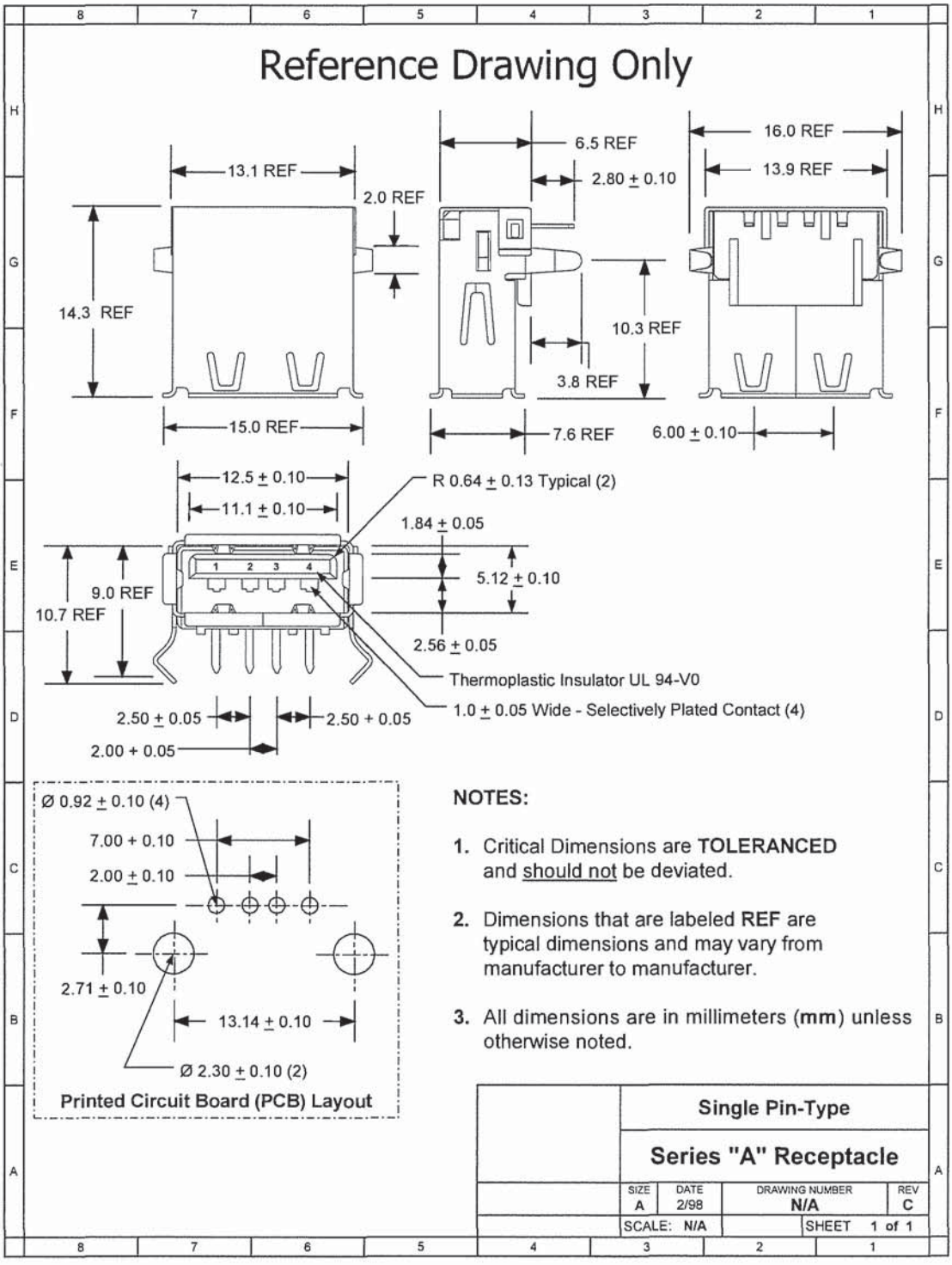


Figure 6-12. Single Pin-type Series "A" Receptacle

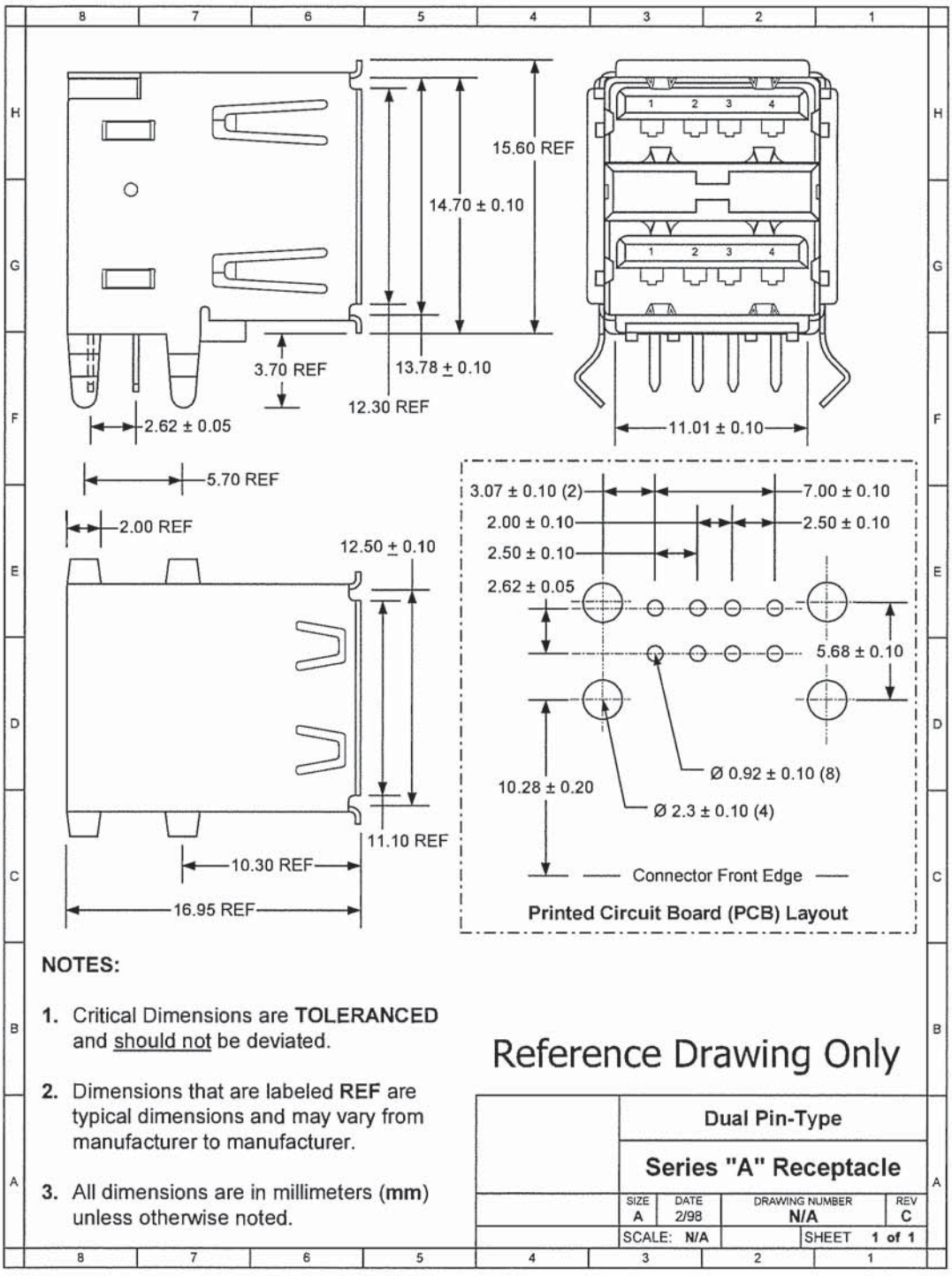


Figure 6-13. Dual Pin-type Series "A" Receptacle

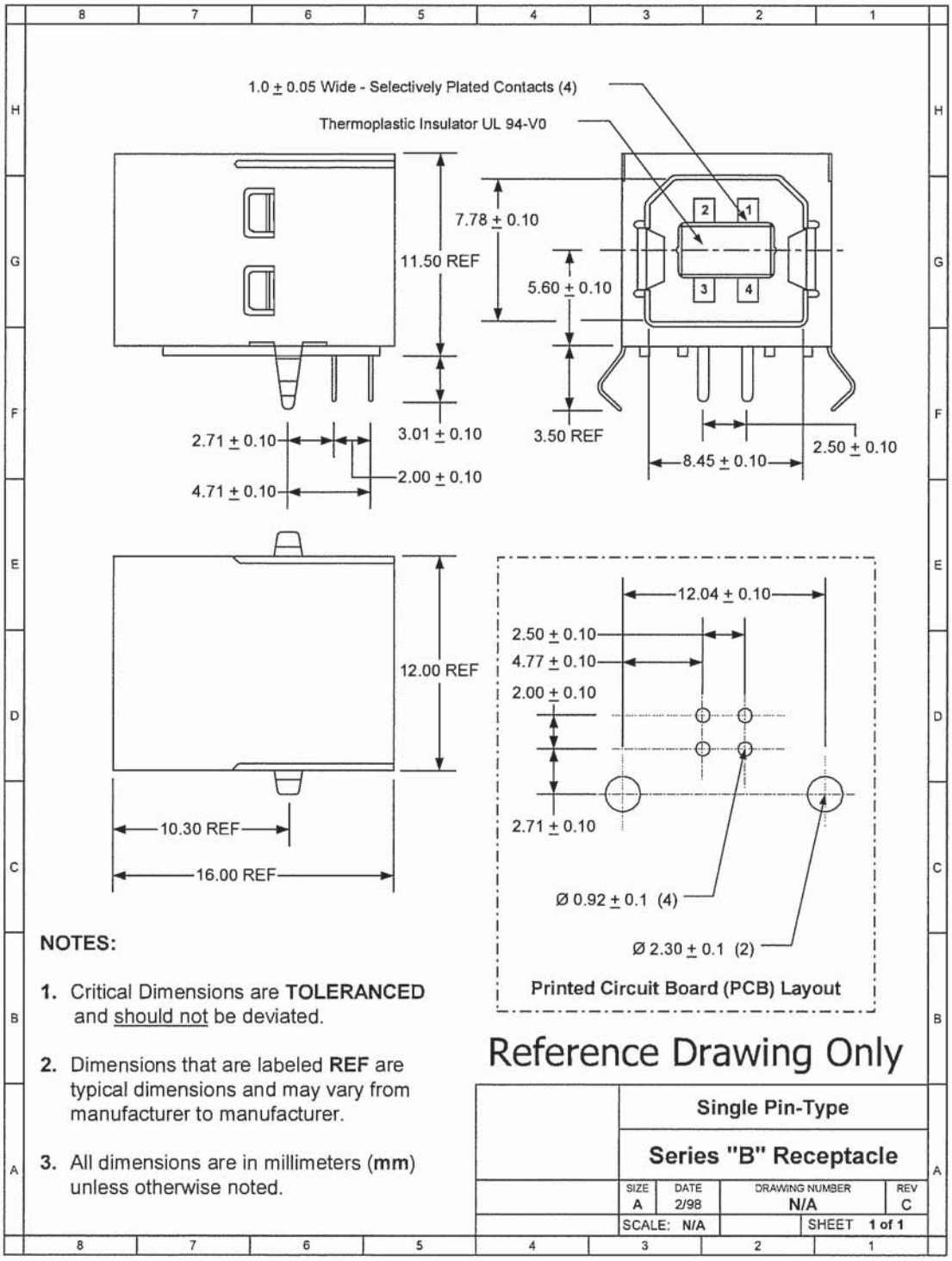


Figure 6-14. Single Pin-type Series "B" Receptacle

Chapter 7
Electrical

This chapter describes the electrical specification for the USB. It contains signaling, power distribution, and physical layer specifications. This specification does not address regulatory compliance. It is the responsibility of product designers to make sure that their designs comply with all applicable regulatory requirements.

The USB 2.0 specification requires hubs to support high-speed mode. USB 2.0 devices are not required to support high-speed mode. A high-speed capable upstream facing transceiver must not support low-speed signaling mode. A USB 2.0 downstream facing transceiver must support high-speed, full-speed, and low-speed modes.

To assure reliable operation at high-speed data rates, this specification requires the use of cables that conform to all current cable specifications.

In this chapter, there are numerous references to strings of J's and K's, or to strings of 1's and 0's. In each of these instances, the leftmost symbol is transmitted/received first, and the rightmost is transmitted/received last.

7.1 Signaling

The signaling specification for the USB is described in the following subsections.

Overview of High-speed Signaling

A high-speed USB connection is made through a shielded, twisted pair cable that conforms to all current USB cable specifications.

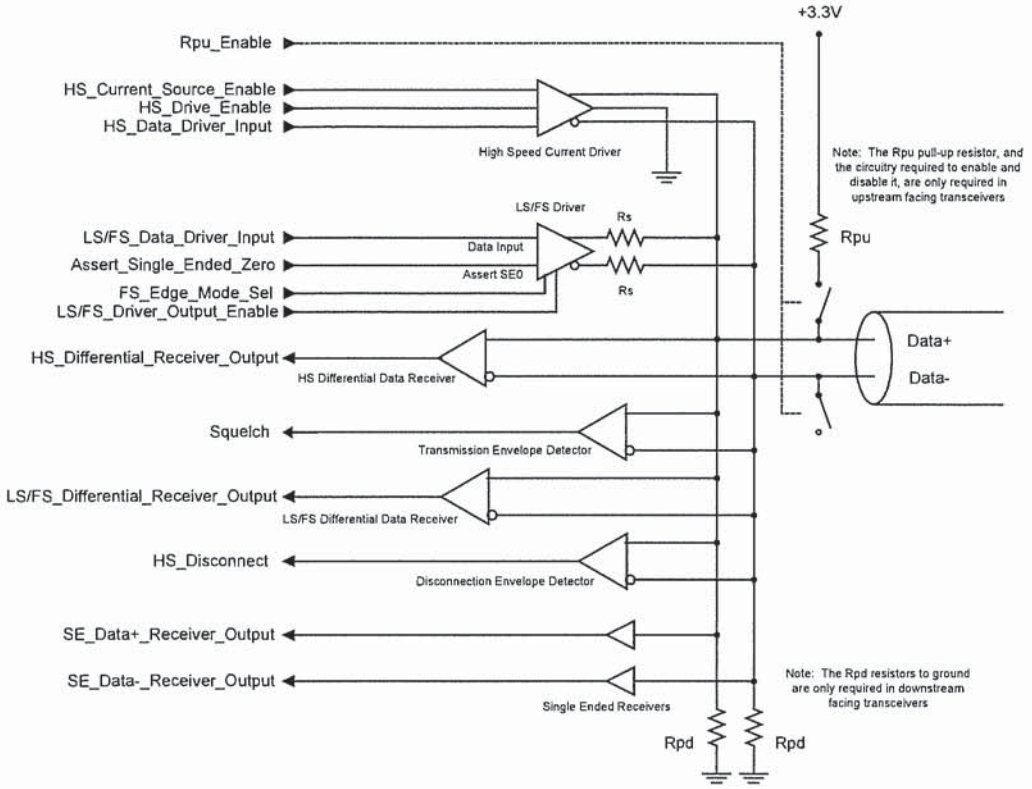


Figure 7-1. Example High-speed Capable Transceiver Circuit

Figure 7-1 depicts an example implementation which largely utilizes USB 1.1 transceiver elements and adds the new elements required for high-speed operation.

High-speed operation supports signaling at 480 Mb/s. To achieve reliable signaling at this rate, the cable is terminated at each end with a resistance from each wire to ground. The value of this resistance (on each wire) is nominally set to 1/2 the specified differential impedance of the cable, or 45 Ω. This presents a differential termination of 90 Ω.

For a link operating in high-speed mode, the high-speed idle state occurs when the transceivers at both ends of the cable present high-speed terminations to ground, and when neither transceiver drives signaling current into the D+ or D- lines. This state is achieved by using the low-/full-speed driver to assert a single ended zero, and to closely control the combined total of the intrinsic driver output impedance and the Rs resistance (to 45 Ω, nominal). The recommended practice is to make the intrinsic driver impedance as low as possible, and to let Rs contribute as much of the 45 Ω as possible. This will generally lead to the best termination accuracy with the least parasitic loading.

In order to transmit in high-speed mode, a transceiver activates an internal current source which is derived from its positive supply voltage and directs this current into one of the two data lines via a high speed current steering switch. In this way, the transceiver generates the high-speed J or K state on the cable.

The dynamic switching of this current into the D+ or D- line follows the same NRZI data encoding scheme used in low-speed or full-speed operation and also in the bit stuffing behavior. To signal a J, the current is directed into the D+ line, and to signal a K, the current is directed into the D- line. The SYNC field and the EOP delimiters have been modified for high-speed mode.

The magnitude of the current source and the value of the termination resistors are controlled to specified tolerances, and together they determine the actual voltage drive levels. The DC resistance from D+ or D- to the device ground is required to be 45 Ω ±10% when measured without a load, and the differential output voltage measured across the lines (in either the J or K state) must be ±400 mV ±10% when D+ and D- are terminated with precision 45 Ω resistors to ground.

The differential voltage developed across the lines is used for three purposes:

- A differential receiver at the receiving end of the cable receives the differential data signal.
- A differential envelope detector at the receiving end of the cable determines when the link is in the Squelch state. A receiver uses squelch detection as indication that the signal at its connector is not valid.
- In the case of a downstream facing hub transceiver, a differential envelope detector monitors whether the signal at its connector is in the high-speed state. A downstream facing transceiver operating in high-speed mode is required to test for this state at a particular point in time when it is transmitting a SOF packet, as described in Section 7.1.7.3. This is used to detect device disconnection. In the absence of the far end terminations, the differential voltage will nominally double (as compared to when a high-speed device is present) when a high-speed J or K are continuously driven for a period exceeding the round-trip delay for the cable and board-traces between the two transceivers.

USB 2.0 requires that a downstream facing transceiver must be able to operate in low-speed, full-speed, and high-speed signaling modes. An upstream facing high-speed capable transceiver must not operate in low-speed signaling mode, but must be able to operate in full-speed signaling mode. Therefore, a 1.5 kΩ pull-up on the D- line is not allowed for a high-speed capable device, since a high-speed capable transceiver must never signal low-speed operation to the hub port to which it is attached.

Table 7-1 describes the required functional elements of a high-speed capable transceiver, using the diagram shown in Figure 7-1 as an example.

Table 7-1. Description of Functional Elements in the Example Shown in Figure 7-1

Element	Description
Low-/full-speed Driver	The low-/full-speed driver is used for low-speed and full-speed transmission. It is required to meet all specifications called out in USB 1.1 for low-speed and full-speed operation, with one exception. The exception is that in high-speed capable transceivers, the impedance of each output, including the contribution of Rs, must be 45 Ω ±10%. The line terminations for high-speed operation are created by having this driver drive D+ and D- to ground. (This is equivalent to driving SE0 in the full-speed or low-speed mode.) Because of the output impedance requirement described above, this provides a well-controlled high-speed termination on each data line to ground. This is equivalent to a 90 Ω differential termination.
Low-/full-speed Differential Receiver	The low-/full-speed differential receiver is used for receiving low-speed and full-speed data.
Single Ended Receivers	The single ended receivers are used for low-speed and full-speed signaling.
High-speed Current Driver	The high-speed current driver is used for high-speed data transmission. A current source derived from a positive supply is switched into either the D+ or D- lines to signal a J or a K, respectively. The nominal value of the current source is 17.78 mA. When this current is applied to a data line with a 45 Ω termination to ground at each end, the nominal high level voltage (V_{HSOH}) is +400 mV. The nominal differential high-speed voltage (D+ - D-) is thus 400 mV for a J and -400 mV for a K. The current source must comply with the Transmit Eye Pattern Templates specified in Section 7.1.2.2, starting with the first symbol of a packet. One means of achieving this is to leave the current source on continuously when a transceiver is operating in high-speed mode. If this approach is used, the current can be directed to the port ground when the transceiver is not transmitting (the example design in Figure 7-1 shows a control line called HS_Current_Source_Enable to turn the current on, and another called HS_Drive_Enable to direct the current into the data lines.) The penalty of this approach is the 17.78 mA of standing current for every such enabled transceiver in the system. The preferred design is to fully turn the current source off when the transceiver is not transmitting.
High-speed Differential Data Receiver	The high-speed differential data receiver is used to receive high-speed data. It is left to transceiver designers to choose between incorporating separate high-speed and low-/full-speed receivers, as shown in Figure 7-1, or combining both functions into a single receiver.

Table 7-1. Description of Functional Elements in the Example Shown in Figure 7-1 (Continued)

Transmission Envelope Detector	This envelope detector is used to indicate that data is invalid when the amplitude of the differential signal at a receiver's inputs falls below the squelch threshold (V_{HSSQ}). It must indicate Squelch when the signal drops below 100 mV differential amplitude, and it must indicate that the line is not in the Squelch state when the signal exceeds 150 mV differential amplitude. The response time of the detector must be fast enough to allow a receiver to detect data transmission, to achieve DLL lock, and to detect the end of the SYNC field within 12 bit times, the minimum number of SYNC bits that a receiver is guaranteed to see. This envelope detector must incorporate a filtering mechanism that prevents indication of squelch during the longest differential data transitions allowed by the receiver eye pattern specifications.
Disconnection Envelope Detector	This envelope detector is required in downstream facing ports to detect the high-speed Disconnect state on the line (V_{HSDSC}). Disconnection must be indicated when the amplitude of the differential signal at the downstream facing driver's connector ≥625 mV, and it must not be indicated when the signal amplitude is ≤525 mV. The output of this detector is sampled at a specific time during the transmission of the high-speed SOF EOP, as described in Section 7.1.7.3.
Pull-up Resistor (R_{PU})	This resistor is required only in upstream facing transceivers and is used to indicate signaling speed capability. A high-speed capable device is required to initially attach as a full-speed device and must transition to high-speed as described in this specification. Once operating in high-speed, the 1.5 kΩ resistor must be electrically removed from the circuit. In Figure 7-1, a control line called R_{PU}_Enable is indicated for this purpose. The preferred embodiment is to attach matched switching devices to both the D+ and D- lines so as to keep the lines' parasitic loading balanced, even though a pull-up resistor must never be used on the D- line of an upstream facing high-speed capable transceiver. When connected, this pull-up must meet all the specifications called out for full-speed operation.
Pull-down Resistors (R_{PD})	These resistors are required only in downstream facing transceivers and must conform to the same specifications called out for low-speed and full-speed operation.

7.1.1 USB Driver Characteristics

The USB uses a differential output driver to drive the USB data signal onto the USB cable.

For low-speed and full-speed operation, the static output swing of the driver in its low state must be below V_{OL} (max) of 0.3 V with a 1.5 kΩ load to 3.6 V, and in its high state must be above the V_{OH} (min) of 2.8 V with a 15 kΩ load to ground as listed in Table 7-7. Full-speed drivers have more stringent requirements, as described in Section 7.1.1.1. The output swings between the differential high and low state must be well-balanced to minimize signal skew. Slew rate control on the driver is required to minimize the radiated noise and cross talk. The driver's outputs must support three-state operation to achieve bi-directional half-duplex operation.

Low-speed and full-speed USB drivers must never "intentionally" generate an SE1 on the bus. SE1 is a state in which both the D+ and D- lines are at a voltage above V_{OSE1} (min), which is 0.8 V.

High-speed drivers use substantially different signaling levels, as described in Section 7.1.1.3.

USB ports must be capable of withstanding continuous exposure to the waveforms shown in Figure 7-2 while in any drive state. These waveforms are applied directly into each USB data pin from a voltage source with an

output impedance of 39 Ω. The open-circuit voltage of the source shown in Figure 7-2 is based on the expected worst-case overshoot and undershoot.

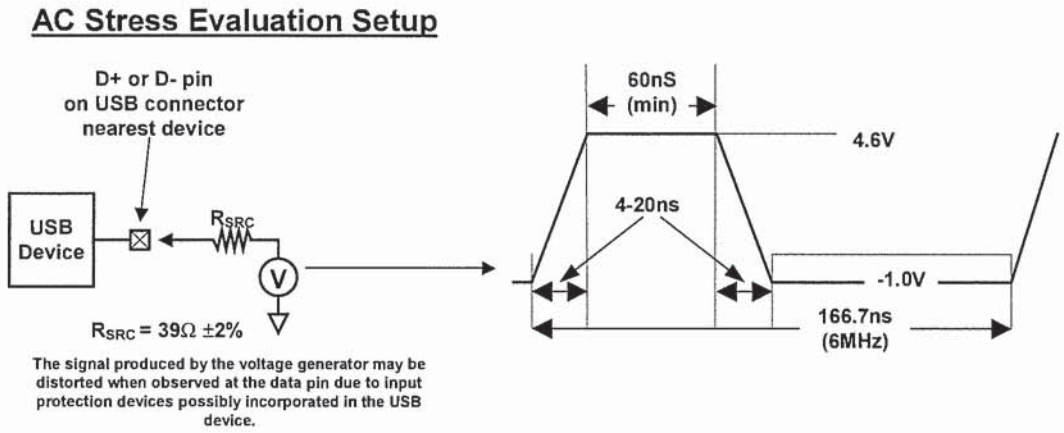


Figure 7-2. Maximum Input Waveforms for USB Signaling

Short Circuit Withstand

A USB transceiver is required to withstand a continuous short circuit of D+ and/or D- to VBUS, GND, other data line, or the cable shield at the connector, for a minimum of 24 hours without degradation. It is recommended that transceivers be designed so as to withstand such short circuits indefinitely. The device must not be damaged under this short circuit condition when transmitting 50% of the time and receiving 50% of the time (in all supported speeds). The transmit phase consists of a symmetrical signal that toggles between drive high and drive low. This requirement must be met for max value of VBUS (5.25 V).

It is recommended that these AC and short circuit stresses be used as qualification criteria against which the long-term reliability of each device is evaluated.

7.1.1.1 Full-speed (12 Mb/s) Driver Characteristics

A full-speed USB connection is made through a shielded, twisted pair cable with a differential characteristic impedance (Z_0) of 90 Ω ±15%, a common mode impedance (Z_{CM}) of 30 Ω ±30%, and a maximum one-way delay (T_{FSCBL}) of 26 ns. When the full-speed driver is not part of a high-speed capable transceiver, the impedance of each of the drivers (Z_{DRV}) must be between 28 Ω and 44 Ω, i.e., within the gray area in Figure 7-4. When the full-speed driver is part of a high-speed capable transceiver, the impedance of each of the drivers (Z_{HSDRV}) must be between 40.5 Ω and 49.5 Ω, i.e., within the gray area in Figure 7-5.

For a CMOS implementation, the driver impedance will typically be realized by a CMOS driver with an impedance significantly less than this resistance with a discrete series resistor making up the balance as shown in Figure 7-3. The series resistor R_S is included in the buffer impedance requirement shown in Figure 7-4 and Figure 7-5. In the rest of the chapter, references to the buffer assume a buffer with the series impedance unless stated otherwise.

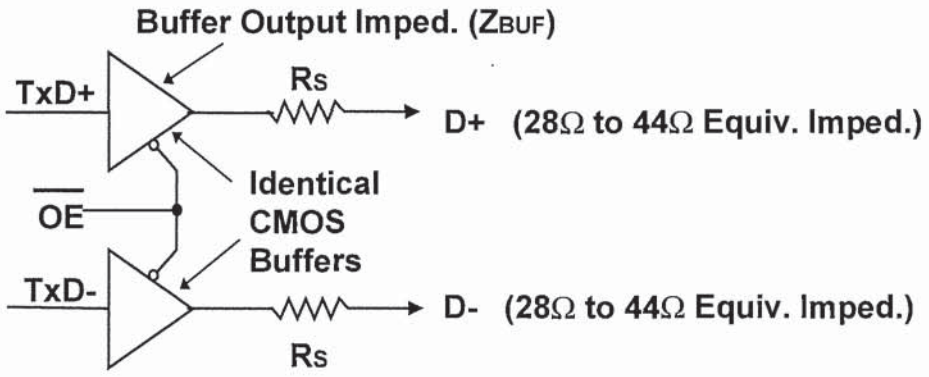


Figure 7-3. Example Full-speed CMOS Driver Circuit (non High-speed capable)

Full-speed Buffers in Transceivers Which are Not High-speed Capable

The buffer impedance must be measured for driving high as well as driving low. Figure 7-4 shows the composite V/I characteristics for the full-speed drivers with included series damping resistor (RS). The characteristics are normalized to the steady-state, unloaded output swing of the driver. The normalized driver characteristics are found by dividing the measured voltages and currents by the actual swing of the driver under test. The normalized V/I curve for the driver must fall entirely inside the shaded region. The V/I region is bounded by the minimum driver impedance above and the maximum driver impedance below. The minimum drive region is intersected by a constant current region of |6.1VOH| mA when driving low and -|6.1VOH| mA when driving high. In the special case of a full-speed driver which is driving low, and which is part of a high-speed capable transceiver, the low drive region is intersected by a constant current region of 22.0 mA. This is the minimum current drive level necessary to ensure that the waveform at the receiver crosses the opposite single-ended switching level on the first reflection.

When testing, the current into or out of the device need not exceed ±10.71*VOH mA and the voltage applied to D+/D- need not exceed 0.3*VOH for the drive low case and need not drop below 0.7*VOH for the drive high case.

Full-speed Buffers in High-speed Capable Transceivers

Figure 7-5 shows the V/I characteristics for a Full-speed buffer which is part of a high-speed capable transceiver. The output impedance, ZHSDRV (including the contribution of RS), is required to be between 40.5 Ω and 49.5 Ω. Additionally, the output voltage must be within 10mV of ground when no current is flowing in or out of the pin (VHSTERM).

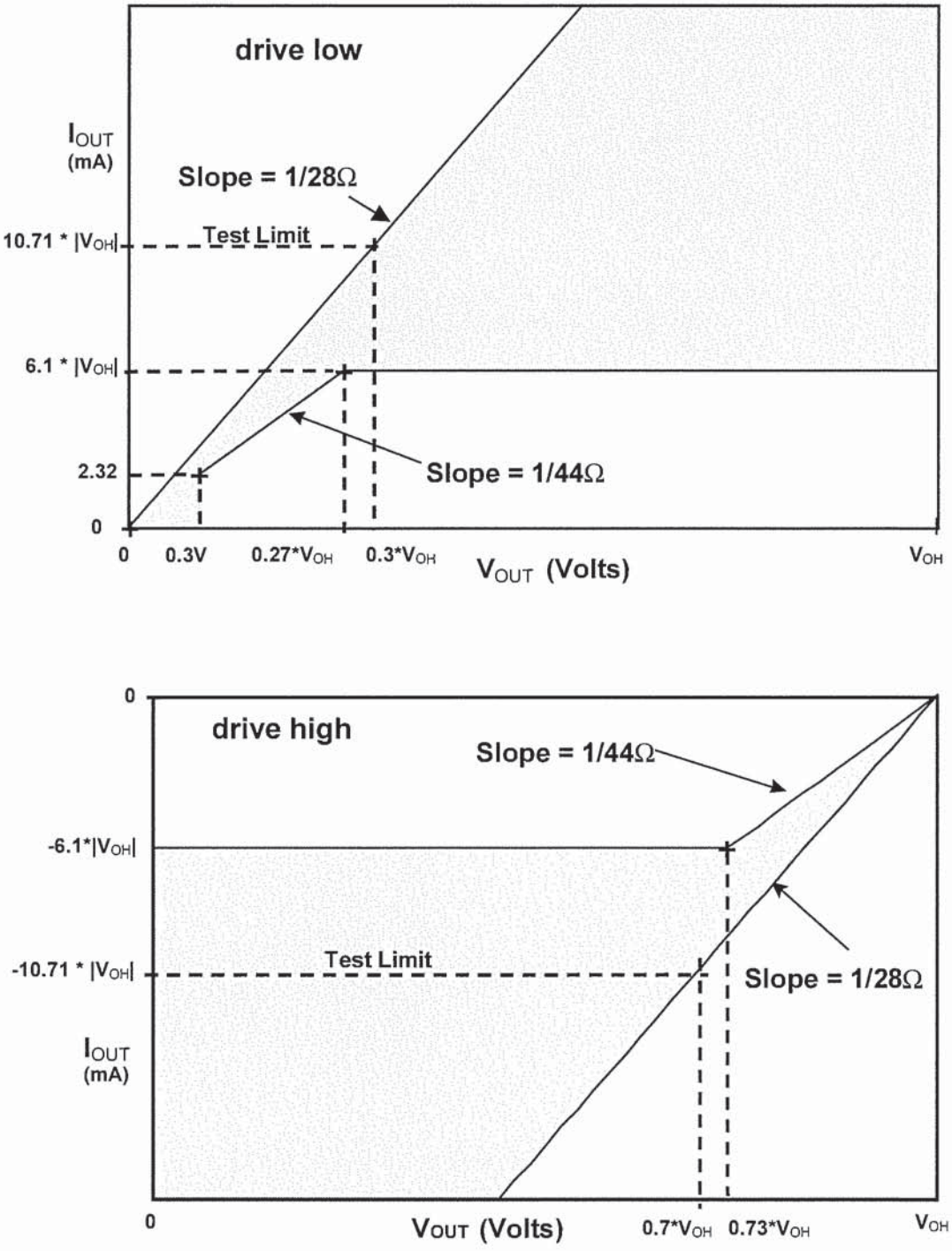


Figure 7-4. Full-speed Buffer V/I Characteristics

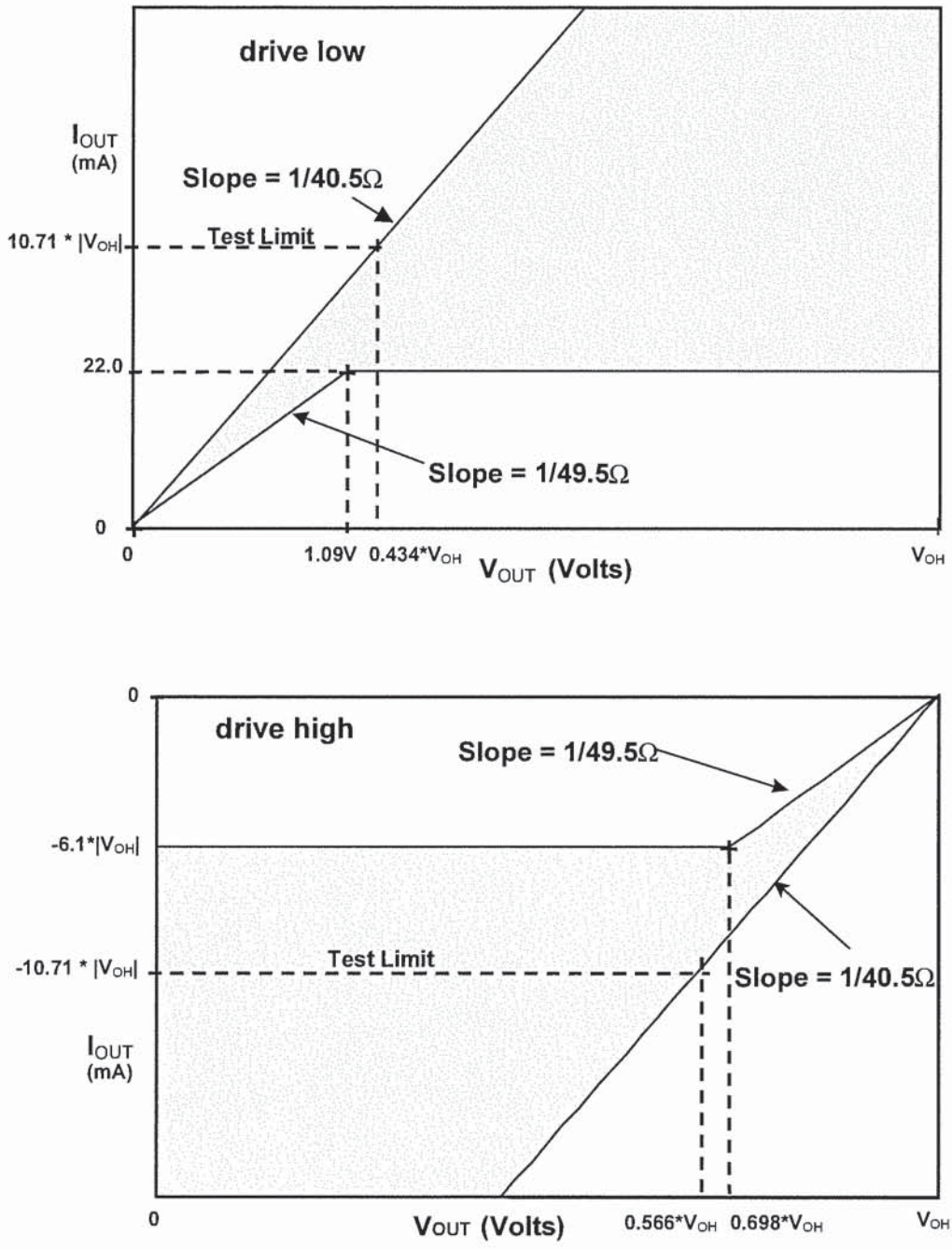


Figure 7-5. Full-speed Buffer V/I Characteristics for High-speed Capable Transceiver

Figure 7-6 shows the full-speed driver signal waveforms.

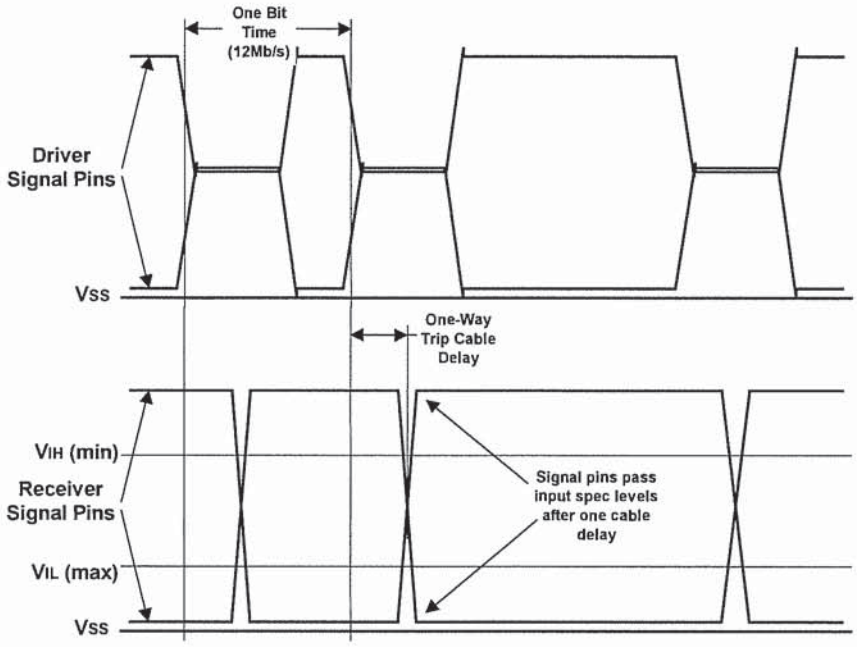


Figure 7-6. Full-speed Signal Waveforms

7.1.1.2 Low-speed (1.5 Mb/s) Driver Characteristics

A low-speed device must have a captive cable with the Series A connector on the plug end. The combination of the cable and the device must have a single-ended capacitance of no less than 200 pF and no more than 450 pF on the D+ or D- lines.

The propagation delay (TLSCBL) of a low-speed cable must be less than 18 ns. This is to ensure that the reflection occurs during the first half of the signal rise/fall, which allows the cable to be approximated by a lumped capacitance.

Figure 7-7 shows the low-speed driver signal waveforms.

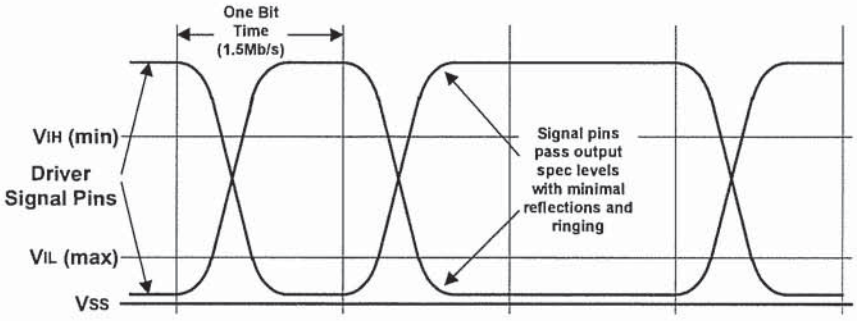


Figure 7-7. Low-speed Driver Signal Waveforms

7.1.1.3 High-speed (480 Mb/s) Driver Characteristics

A high-speed USB connection is made through a shielded, twisted pair cable with a differential characteristic impedance (Z_0) of 90 Ω ±15%, a common mode impedance (Z_{CM}) of 30 Ω ±30%, and a maximum one-way delay of 26 ns (T_{FSCBL}). The D+ and D- circuit board traces which run between a transceiver and its associated connector should also have a nominal differential impedance of 90 Ω, and together they may add an additional 4 ns of delay between the transceivers. (See Section 7.1.6 for details on impedance specifications of boards and transceivers.) The differential output impedance of a high-speed capable driver is required to be 90 Ω ±10%. When either the D+ or D- lines are driven high, V_{HSOH} (the high-speed mode high-level output voltage driven on a data line with a precision 45 Ω load to GND) must be 400 mV ±10%. On a line which is not driven, either because the transceiver is not transmitting or because the opposite line is being driven high, V_{HSOL} (the high-speed mode low-level output voltage driven on a data line with a 45 Ω load to GND) must be 0 V ± 10 mV.

Note: Unless indicated otherwise, all voltage measurements are to be made with respect to the local circuit ground.

Note: This specification requires that a high-speed capable transceiver operating in full-speed or low-speed mode must have a driver impedance (Z_{HSDRV}) of 45 Ω ±10%. It is recommended that the driver impedances be matched to within 5 Ω within a transceiver. For upstream facing transceivers which do not support high-speed mode, the driver output impedance (Z_{DRV}) must fall within the range of 28 Ω to 44 Ω.

On downstream facing ports, R_{PD} resistors (15 kΩ ±5%) must be connected from D+ and D- to ground.

When a high-speed capable transceiver transitions to high-speed mode, the high-speed idle state is achieved by driving SE0 with the low-/full-speed drivers at each end of the link (so as to provide the required terminations), and by disconnecting the D+ pull-up resistor in the upstream facing transceiver.

In the preferred embodiment, a transceiver activates its high-speed current driver only when transmitting high-speed signals. This is a potential design challenge, however, since the signal amplitude and timing specifications must be met even on the first symbol within a packet. As a less efficient alternative, a transceiver may cause its high-speed current source to be continually active while in high-speed mode. When the transceiver is not transmitting, the current may be directed into the device ground rather than through the current steering switch which is used for data signaling. In the example circuit, steering the current to ground is accomplished by setting HS_Drive_Enable low.

In CMOS implementations, the driver impedance will typically be realized by the combination of the driver's intrinsic output impedance and Rs. To optimally control Z_{HSDRV} and to minimize parasitics, it is preferred the driver impedance be minimized (under 5 Ω) and the balance of the 45 Ω should be contributed by the Rs component.

When a transceiver operating in high-speed mode transmits, the transmit current is directed into either the D+ or D- data line. A J is asserted by directing the current to the D+ line, a K by directing it to the D- line.

When each of the data lines is terminated with a 45 Ω resistor to the device ground, the effective load resistance on each side is 22.5 Ω. Therefore, the line into which the drive current is being directed rises to 17.78 ma * 22.5 Ω or 400 mV (nominal). The other line remains at the device ground voltage. When the current is directed to the opposite line, these voltages are reversed.

7.1.2 Data Signal Rise and Fall, Eye Patterns

The following sections specify the data signal rise and fall times for full-speed and low-speed signaling, and the rise time and eye patterns for high-speed signaling.

7.1.2.1 Low-speed and Full-speed Data Signal Rise and Fall

For low-speed and full-speed, the output rise time and fall times are measured between 10% and 90% of the signal (Figure 7-8). Rise and fall time requirements apply to differential transitions as well as to transitions between differential and single-ended signaling.

The rise and fall times for full-speed buffers are measured with the load shown in Figure 7-9. The rise and fall times must be between 4 ns and 20 ns and matched to within ±10% to minimize RFI emissions and signal skew. The transitions must be monotonic.

The rise and fall times for low-speed buffers are measured with the load shown in Figure 7-10. The capacitive load shown in Figure 7-10 is representative of the worst-case load allowed by the specification. A downstream facing transceiver is allowed 150 pF of input/output capacitance (CIND). A low-speed device (including cable) may have a capacitance of as little as 200 pF and as much as 450 pF. This gives a range of 200 pF to 600 pF as the capacitive load that a downstream facing low-speed buffer might encounter. Upstream facing buffers on low-speed devices must be designed to drive the capacitance of the attached cable plus an additional 150 pF. If a low-speed buffer is designed for an application where the load capacitance is known to fall in a different range, the test load can be adjusted to match the actual application. Low-speed buffers on hosts and hubs that are attached to USB receptacles must be designed for the 200 pF to 600 pF range. The rise and fall time must be between 75 ns and 300 ns for any balanced, capacitive test load. In all cases, the edges must be matched to within ±20% to minimize RFI emissions and signal skew. The transitions must be monotonic.

For both full-speed and low-speed signaling, the crossover voltage (VCRS) must be between 1.3 V and 2.0 V.

For low-speed and full-speed, this specification does not require matching signal swing matching to any greater degree than described above. However, when signaling, it is preferred that the average voltage on the D+ and D- lines should be constant. This means that the amplitude of the signal swing on both D+ and D- should be the same; the low and high going transition should begin at the same time and change at the same rate; and the crossover voltage should be the same when switching to a J or K. Deviations from signal matching will result in common-mode noise that will radiate and affect the ability of devices and systems to pass tests that are mandated by government agencies.

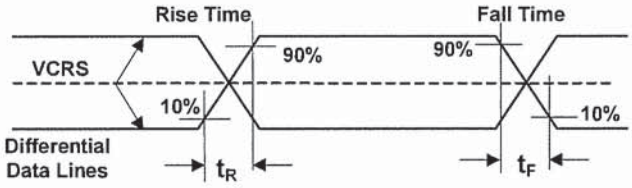


Figure 7-8. Data Signal Rise and Fall Time

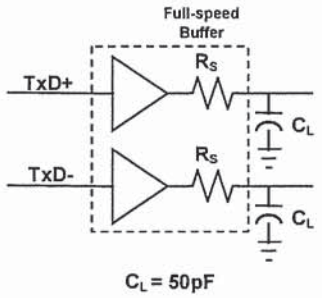


Figure 7-9. Full-speed Load

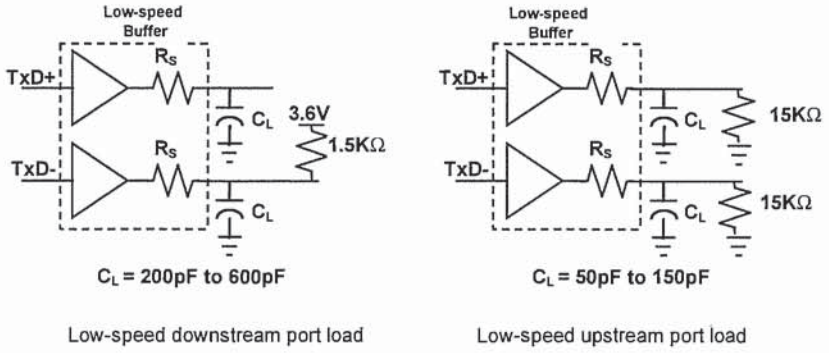


Figure 7-10. Low-speed Port Loads

Note: The CL for low-speed port load only represents the range of loading that might be added when the low-speed device is attached to a hub. The low-speed buffer must be designed to drive the load of its attached cable plus CL. A low-speed buffer design that can drive the downstream test load would be capable of driving any legitimate upstream load.

7.1.2.2 High-speed Signaling Eye Patterns and Rise and Fall Time

The following specifications apply to high-speed mode signaling. All bits, including the first and last bit of a packet, must meet the following eye pattern requirements for timing and amplitude.

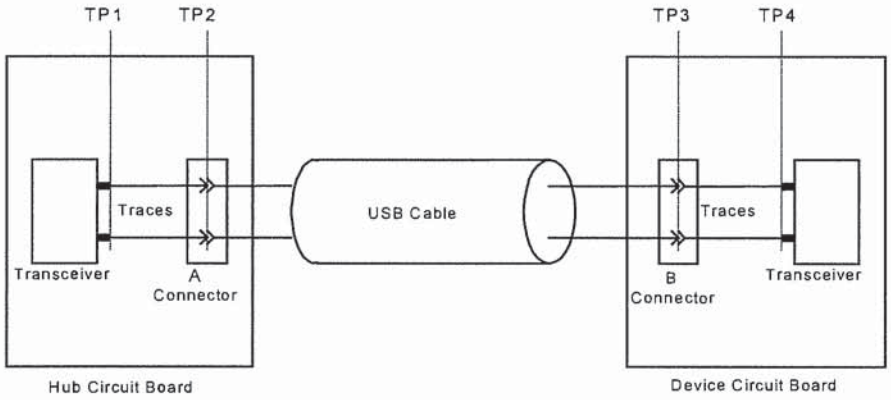


Figure 7-11. Measurement Planes

Figure 7-11 defines four test planes which will be referenced in this section. TP1 and TP4 are the points where the transceiver IC pins are soldered to the hub and device circuit boards, respectively. TP2 is at the mated pins of the A connector, and TP3 is at the mated pins of the B connector (or, in the case of a captive cable, where the cable is attached to the circuit board). The following differential eye pattern templates specify transmit waveform and receive sensitivity requirements at various points and under various conditions.

When testing high-speed transmitters and receivers, measurements are made with the Transmitter/Receiver Test Fixture shown in Figure 7-12. In either case, the fixture is attached to the USB connector closest to the transceiver being tested.

Transmitter Test Attenuation: Voltage at Scope Inputs = 0.760 * Voltage at Transmitter Outputs
Receiver Test Attenuation: Voltage at Receiver Inputs = 0.684 * Voltage at Data Generator Outputs

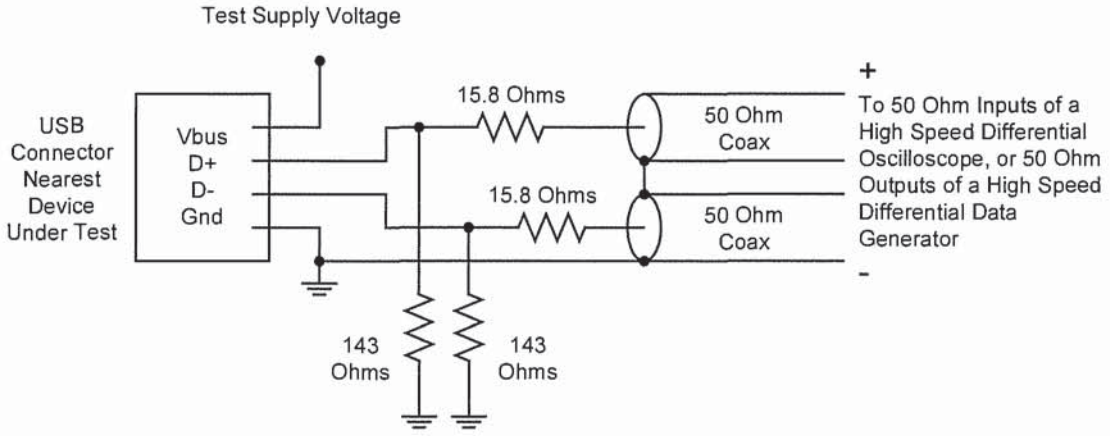


Figure 7-12. Transmitter/Receiver Test Fixture

Note: When testing the upstream facing port of a device, VBUS must be provided from the time the device is placed in the appropriate test mode until the test is completed. This requirement will likely necessitate additional switching functionality in the test fixture (for example, to switch the D+ and D- lines between the host controller and the test instrument). Such additions must have minimal impact on the high frequency measurement results.

Transmit eye patterns specify the minimum and maximum limits, as well as limits on timing jitter, within which a driver must drive signals at each of the specified test planes. Receive eye patterns specify the minimum and maximum limits, as well as limits on timing jitter, within which a receiver must recover data.

Conformance to Templates 1, 2, 3, and 4 is required for USB 2.0 hubs and devices:

Template 1: Transmit waveform requirements for hub measured at TP2, and for device (without a captive cable) measured at TP3

Template 2: Transmit waveform requirements for device (with a captive cable) measured at TP2

Template 3: Receiver sensitivity requirements for device (with a captive cable) when signal is applied at TP2

Template 4: Receiver sensitivity requirements for device (without a captive cable) when signal is applied at TP3, and for hub when signal is applied at TP2

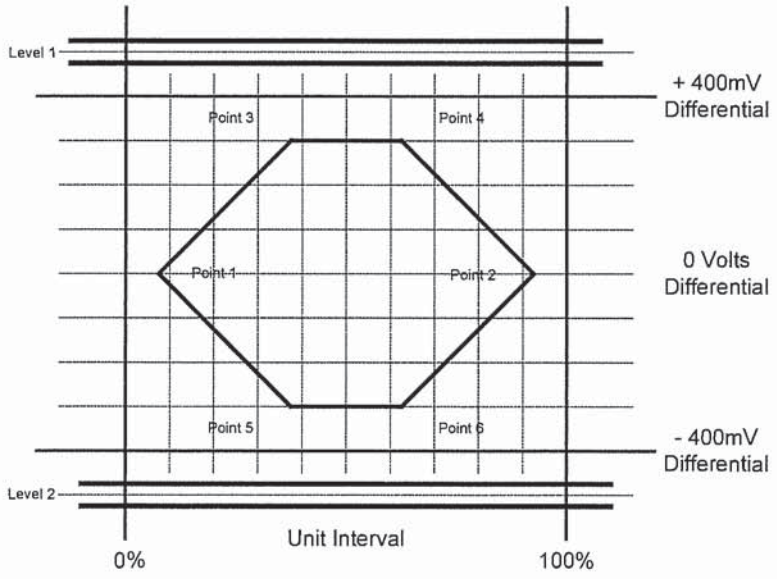
Templates 5 and 6 are recommended guidelines for designers:

Template 5: Transmit waveform requirements for hub transceiver measured at TP1, and for device transceiver measured at TP4

Template 6: Receiver sensitivity requirements for device transceiver when signal is applied at TP4, and for hub transceiver at when signal is applied at TP1

Template 1

Figure 7-13 shows the transmit waveform requirements for a hub measured at TP2, and for a device (without a captive cable) measured at TP3.

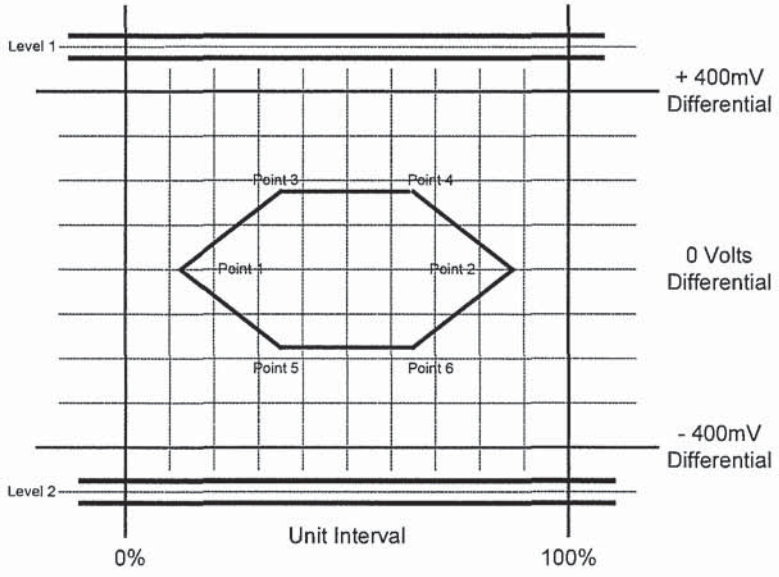


	Voltage Level (D+ - D-)	Time (% of Unit Interval)
Level 1	525 mV in UI following a transition, 475 mV in all others	N/A
Level 2	-525 mV in UI following a transition, -475 in all others	N/A
Point 1	0 V	7.5% UI
Point 2	0 V	92.5% UI
Point 3	300 mV	37.5% UI
Point 4	300 mV	62.5% UI
Point 5	-300 mV	37.5% UI
Point 6	-300 mV	62.5% UI

Figure 7-13. Template 1

Template 2

Figure 7-14 shows transmit waveform requirements for a device (with a captive cable) measured at TP2.

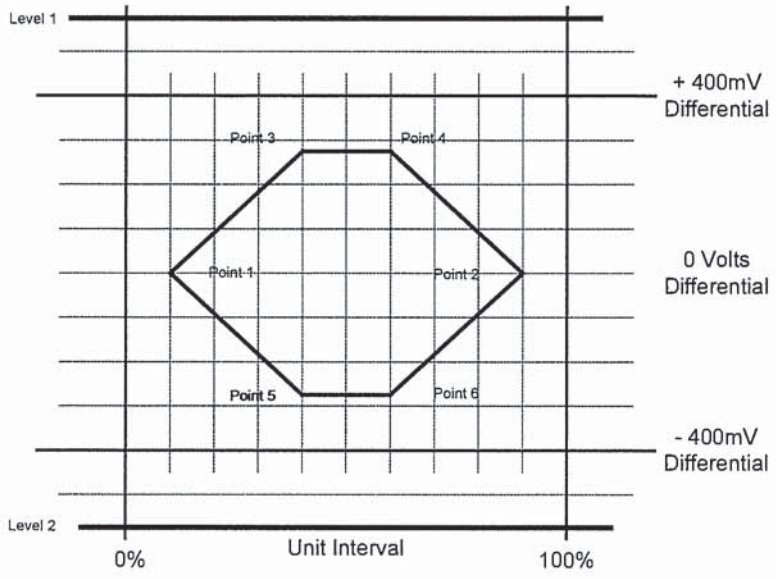


	Voltage Level (D+ - D-)	Time (% of Unit Interval)
Level 1	525 mV in UI following a transition, 475 mV in all others	N/A
Level 2	-525 mV in UI following a transition, -475 in all others	N/A
Point 1	0 V	12.5% UI
Point 2	0 V	87.5% UI
Point 3	175 mV	35% UI
Point 4	175 mV	65% UI
Point 5	-175 mV	35% UI
Point 6	-175 mV	65% UI

Figure 7-14. Template 2

Template 3

Figure 7-15 shows receiver sensitivity requirements for a device (with a captive cable) when a signal is applied at TP2.



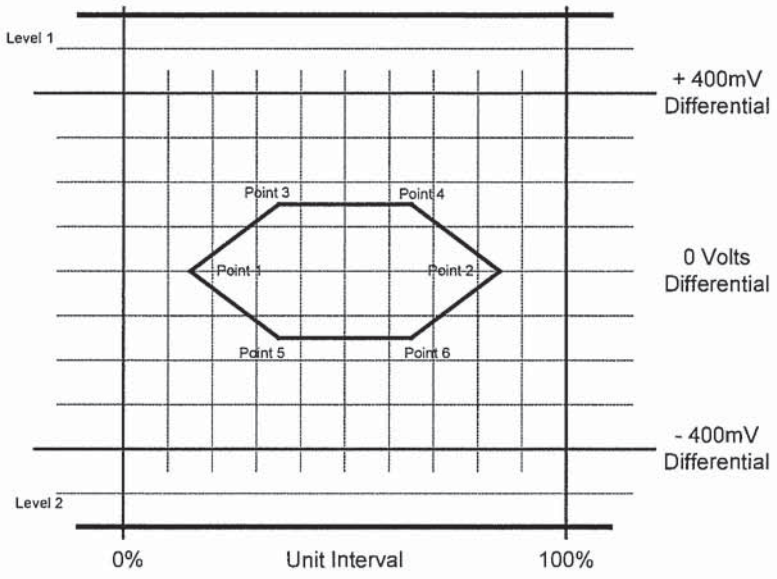
	Voltage Level (D+ - D-)	Time (% of Unit Interval)
Level 1	575 mV	N/A
Level 2	-575 mV	N/A
Point 1	0 V	10% UI
Point 2	0 V	90% UI
Point 3	275 mV	40% UI
Point 4	275 mV	60% UI
Point 5	-275 mV	40% UI
Point 6	-275 mV	60% UI

Figure 7-15. Template 3

Note: This eye is intended to specify differential data receiver sensitivity requirements. Levels 1 and 2 are outside the Disconnect Threshold values, but disconnection is detected at the source (after a minimum of 32 bit times without any transitions), not at the target receiver.

Template 4

Figure 7-16 shows receiver sensitivity requirements for a device (without a captive cable) when signal is applied at TP3, and for a hub when a signal is applied at TP2.



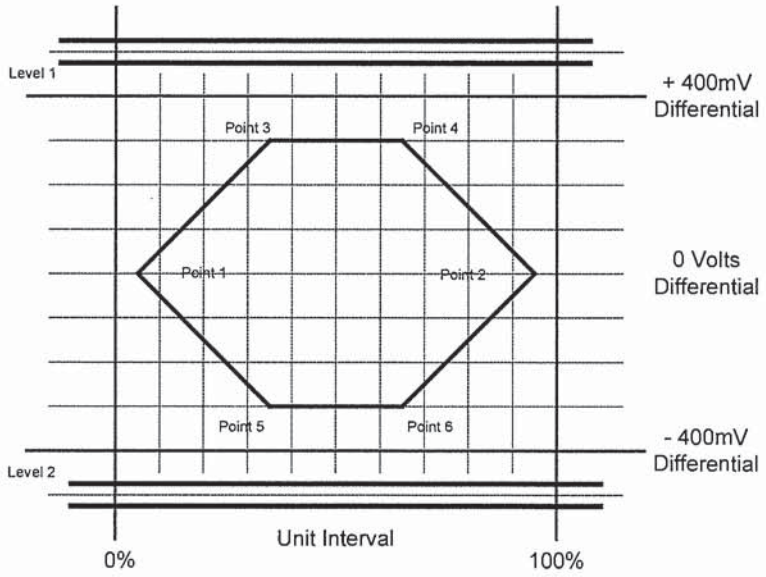
	Voltage Level (D+ - D-)	Time (% of Unit Interval)
Level 1	575 mV	N/A
Level 2	-575 mV	N/A
Point 1	0 V	15% UI
Point 2	0 V	85% UI
Point 3	150 mV	35% UI
Point 4	150 mV	65% UI
Point 5	-150 mV	35% UI
Point 6	-150 mV	65% UI

Figure 7-16. Template 4

Note: This eye is intended to specify differential data receiver sensitivity requirements. Levels 1 and 2 are outside the Disconnect Threshold values, but disconnection is detected at the source (after a minimum of 32 bit times without any transitions), not at the target receiver.

Template 5

Figure 7-17 shows transmit waveform requirements for a hub transceiver measured at TP1 and for a device transceiver measured at TP4.

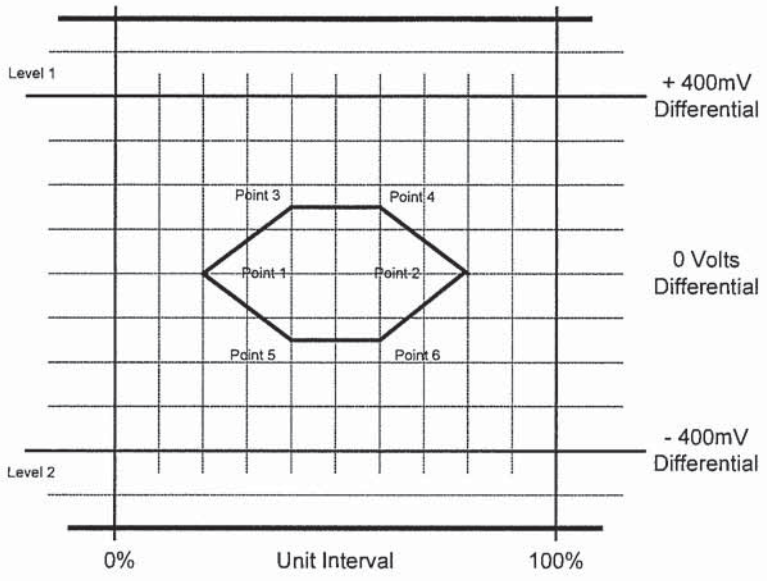


	Voltage Level (D+ - D-)	Time (% of Unit Interval)
Level 1	525 mV in UI following a transition, 475 mV in all others	N/A
Level 2	-525 mV in UI following a transition, -475 in all others	N/A
Point 1	0 V	5% UI
Point 2	0 V	95% UI
Point 3	300 mV	35% UI
Point 4	300 mV	65% UI
Point 5	-300 mV	35% UI
Point 6	-300 mV	65% UI

Figure 7-17. Template 5

Template 6

Figure 7-18 shows receiver sensitivity requirements for a device transceiver when a signal is applied at TP4 and for a hub transceiver when a signal is applied at TP1.



	Voltage Level (D+ - D-)	Time (% of Unit Interval)
Level 1	575 mV	N/A
Level 2	-575 mV	N/A
Point 1	0 V	20% UI
Point 2	0 V	80% UI
Point 3	150 mV	40% UI
Point 4	150 mV	60% UI
Point 5	-150 mV	40% UI
Point 6	-150 mV	60% UI

Figure 7-18. Template 6

Note: This eye is intended to specify differential data receiver sensitivity requirements. Levels 1 and 2 are outside the Disconnect Threshold values, but disconnection is detected at the source (after a minimum of 32 bit times without any transitions), not at the target receiver.

High-speed Signaling Rise and Fall Times

The transition time of a high-speed driver must not be less than the specified minimum allowable differential rise and fall time (T_{HSR} and T_{HSF}). Transition times are measured when driving a reference load of 45 Ω to ground on D+ and D-. Figure 7-12 shows a recommended "Transmitter Test Fixture" for performing these measurements.

For a hub, or for a device with detachable cable, the 10% to 90% high-speed differential rise and fall times must be 500 ps or longer when measured at the A or B receptacles (respectively).

For a device with a captive cable assembly, it is a recommended design guideline that the 10% to 90% high-speed differential rise and fall times must be 500 ps or longer when measured at the point where the cable is attached to the device circuit board.

It is required that high-speed data transitions be monotonic over the minimum vertical openings specified in the preceding eye pattern templates.

7.1.2.3 Driver Usage

The upstream facing ports of functions must use one and only one of the following three driver configurations:

1. Low-speed – Low-speed drivers only
2. Full-speed – Full-speed drivers only
3. Full-/high-speed – Combination full-speed and high-speed drivers

Upstream facing USB 2.0 hub ports must use full-/high-speed drivers. Such ports must be capable of transmitting data at low-speed and full-speed rates with full-speed signaling, and at the high-speed rate using high-speed signaling. Downstream facing ports (including the host) must support low-speed, full-speed, and high-speed signaling, and must be able to transmit data at each of the three associated data rates.

In this section, there is reference to a situation in which high-speed operation is "disallowed." This topic is discussed in depth in Chapter 11 of this specification. In brief, a high-speed capable hub's downstream facing ports are "high-speed disallowed" if the hub is unable to establish a high-speed connection on its upstream facing port. For example, this would be the case for the downstream facing ports of a high-speed capable hub when the hub is connected to a USB 1.1 host controller.

When a full-/high-speed device is attached to a pre-USB 2.0 hub, or to a hub port which is high-speed disallowed, it is required to behave as a full-speed only device. When a full-/high-speed device is attached to a USB 2.0 hub which is not high-speed disallowed, it must operate with high-speed signaling and data rate.

7.1.3 Cable Skew

The maximum skew introduced by the cable between the differential signaling pair (i.e., D+ and D- (T_{SKEW})) must be less than 100 ps and is measured as described in Section 6.7.

7.1.4 Receiver Characteristics

This section discusses the receiver characteristics for low-speed, full-speed, and full-/high-speed transceivers.

7.1.4.1 Low-speed and Full-speed Receiver Characteristics

A differential input receiver must be used to accept the USB data signal. The receiver must feature an input sensitivity (V_{DI}) of at least 200 mV when both differential data inputs are in the differential common mode range (V_{CM}) of 0.8 V to 2.5 V, as shown in Figure 7-19.

In addition to the differential receiver, there must be a single-ended receiver for each of the two data lines. The receivers must have a switching threshold between 0.8 V (V_{IL}) and 2.0 V (V_{IH}). It is recommended that the single-ended receivers incorporate hysteresis to reduce their sensitivity to noise.

Both D+ and D- may temporarily be less than V_{IH} (min) during differential signal transitions. This period can be up to 14 ns (T_{FST}) for full-speed transitions and up to 210 ns (T_{LST}) for low-speed transitions. Logic in the receiver must ensure that that this is not interpreted as an SE0.

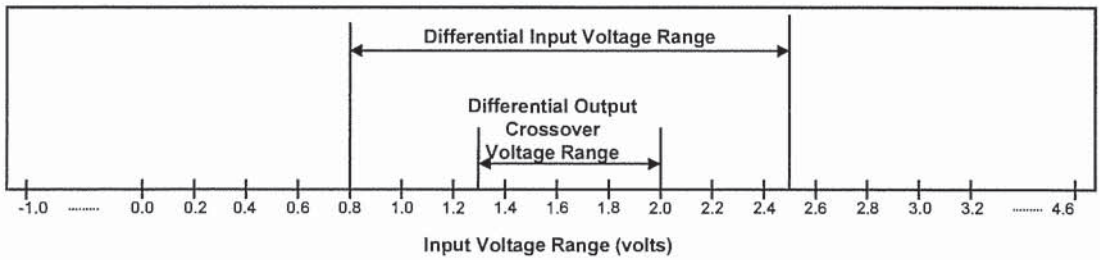


Figure 7-19. Differential Input Sensitivity Range for Low-/full-speed

7.1.4.2 High-speed Receiver Characteristics

A high-speed capable transceiver receiver must conform to the receiver characteristics specifications called out in Section 7.1.4.1 when receiving in low-speed or full-speed modes.

As shown in Figure 7-1, a high-speed capable transceiver which is operating in high-speed mode "listens" for an incoming serial data stream with the high-speed differential data receiver and the transmission envelope detector. Additionally, a downstream facing high-speed capable transceiver monitors the amplitude of the differential voltage on the lines with the disconnection envelope detector.

When receiving in high-speed mode, the differential receiver must be able to reliably receive signals that conform to the Receiver Eye Pattern templates shown in Section 7.1.2. Additionally, it is a strongly recommended guideline that a high-speed receiver should be able to reliably receive such signals in the presence of a common mode voltage component (V_{HSCM}) over the range of –50 mV to 500 mV (the nominal common mode component of high-speed signaling is 200 mV). Low frequency chirp J and K signaling, which occurs during the Reset handshake, should be reliably received with a common mode voltage range of –50 mV to 600 mV.

Reception of data is qualified by the output of the transmission envelope detector. The receiver must disable data recovery when the signal falls below the high-speed squelch level (V_{HSSQ}) defined in Table 7-3. (Detector must indicate squelch when the magnitude of the differential voltage envelope is ≤ 100 mV, and must not indicate squelch if the amplitude of differential voltage envelope is ≥ 150 mV.) Squelch detection must be done with a differential envelope detector, such as the one shown in Figure 7-1. The envelope detector used to detect the squelch state must incorporate a filtering mechanism that prevents indication of squelch during differential data crossovers.

The definition of a high-speed packet's SYNC pattern, together with the requirements for high-speed hub repeaters, guarantee that a receiver will see at least 12 bits of SYNC (KJKJKJKJKJKK) followed by the data portion of the packet. This means that the combination of squelch response time, DLL lock time, and end of SYNC detection must occur within 12 bit times. This is required to assure that the first bit of the packet payload will be received correctly.

In the case of a downstream facing port, a high-speed capable transceiver must include a differential envelope detector that indicates when the signal on the data exceeds the high-speed Disconnect level (V_{HSDSC}) as defined in Table 7-3. (The detector must not indicate that the disconnection threshold has been exceeded if the differential signal amplitude is ≤525 mV, and must indicate that the threshold has been exceeded if the differential signal amplitude is ≥625 mV.)

When sampled at the appropriate time, this detector provides indication that the device has been disconnected. The details of how the disconnection envelope detector is used are described in Section 7.1.7.3.

7.1.5 Device Speed Identification

The following sections specify the speed identification mechanisms for low-speed, full-speed, and high-speed.

7.1.5.1 Low-/Full-speed Device Speed Identification

The USB is terminated at the hub and function ends as shown in Figure 7-20 and Figure 7-21. Full-speed and low-speed devices are differentiated by the position of the pull-up resistor on the downstream end of the cable:

- Full-speed devices are terminated as shown in Figure 7-20 with the pull-up resistor on the D+ line.
- Low-speed devices are terminated as shown in Figure 7-21 with the pull-up resistor on the D- line.
- The pull-down terminators on downstream facing ports are resistors of 15 kΩ ±5% connected to ground.

The design of the pull-up resistor must ensure that the signal levels satisfy the requirements specified in Table 7-2. In order to facilitate bus state evaluation that may be performed at the end of a reset, the design must be able to pull-up D+ or D- from 0 V to VIH (min) within the minimum reset relaxation time of 2.5 µs. A device that has a detachable cable must use a 1.5 kΩ ±5% resistor tied to a voltage source between 3.0 V and 3.6 V (VTERM) to satisfy these requirements. Devices with captive cables may use alternative termination means. However, the Thevenin resistance of any termination must be no less than 900 Ω.

Note: Thevenin resistance of termination does not include the 15 kΩ ±5% resistor on host/hub.

The voltage source on the pull-up resistor must be derived from or controlled by the power supplied on the USB cable such that when VBUS is removed, the pull-up resistor does not supply current on the data line to which it is attached.

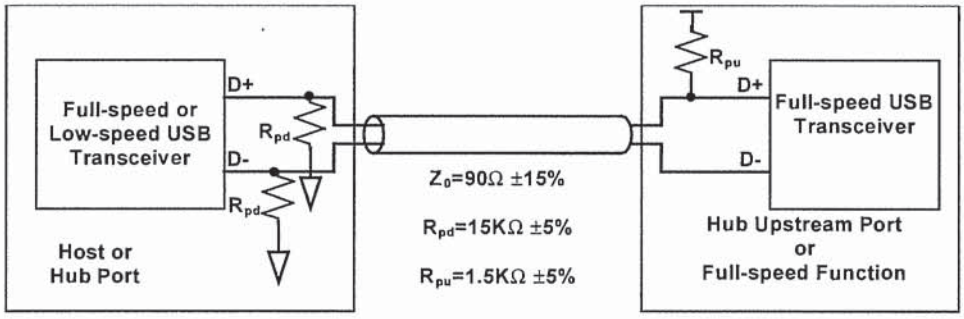


Figure 7-20. Full-speed Device Cable and Resistor Connections

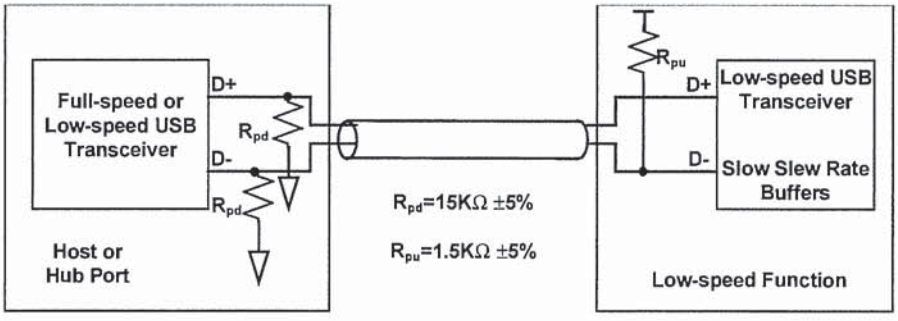


Figure 7-21. Low-speed Device Cable and Resistor Connections

7.1.5.2 High-speed Device Speed Identification

The high-speed Reset and Detection mechanisms follow the behavioral model for low-/full-speed. When reset is complete, the link must be operating in its appropriate signaling mode (low-speed, full-speed, or high-speed as governed by the preceding usage rules), and the speed indication bits in the port status register will correctly report this mode. Software need only initiate the assertion of reset and read the port status register upon notification of reset completion.

High-speed capable devices initially attach as full-speed devices. This means that for high-speed capable upstream facing ports, R_{PU} (1.5 kΩ ±5%) must be connected from D+ to the 3.3 V supply (as shown in Figure 7-1) through a switch which can be opened under SW control.

After the initial attachment, high-speed capable transceivers engage in a low level protocol during reset to establish a high-speed link and to indicate high-speed operation in the appropriate port status register. This protocol is described in Section 7.1.7.5.

7.1.6 Input Characteristics

The following sections describe the input characteristics for transceivers operating in low-speed, full-speed, and high-speed modes.

7.1.6.1 Low-speed and Full-speed Input Characteristics

The input impedance of D+ or D- without termination should be > 300 kΩ (Z_{INP}). The input capacitance of a port is measured at the connector pins. Upstream facing and downstream facing ports are allowed different values of capacitance. The maximum capacitance (differential or single-ended) (C_{IND}) allowed on a downstream facing port of a hub or host is 150 pF on D+ or D- when operating in low-speed or full-speed. This is comprised of up to 75 pF of lumped capacitance to ground on each line at the transceiver and in the connector, and an additional 75 pF capacitance on each conductor in the transmission line between the receptacle and the transceiver. The transmission line between the receptacle and R_S must be 90 Ω ±15%.

The maximum capacitance on an upstream facing port of a full-speed device with a detachable cable (C_{INUB}) is 100 pF on D+ or D-. This is comprised of up to 75 pF of lumped capacitance to ground on each line at the transceiver and in the connector and an additional 25 pF capacitance on each conductor in the transmission line between the receptacle and the transceiver. The difference in capacitance between D+ and D- must be less than 10%.

For full-speed devices with captive cables, the device itself may have up to 75 pF of lumped capacitance to ground on D+ and D-. The cable accounts for the remainder of the input capacitance.

A low-speed device is required to have a captive cable. The input capacitance of the low-speed device will include the cable. The maximum single-ended or differential input capacitance of a low-speed device is 450 pF (C_{LINUA}).

For devices with captive cables, the single-ended input capacitance must be consistent with the termination scheme used. The termination must be able to charge the D+ or D- line from 0 V to V_{IH} (min) within 2.5 µs. The capacitance on D+/D- includes the single-ended input-capacitance of the device (measured from the pins on the connector on the cable) and the 150 pF of input capacitance of the host/hub.

An implementation may use small capacitors at the transceiver for purposes of edge rate control. The sum of the capacitance of the added capacitor (C_{EDGE}), the transceiver, and the trace connecting capacitor and transceiver to R_S must not exceed 75 pF (either single-ended or differential) and the capacitance must be balanced to within 10%. The added capacitor, if present, must be placed between the transceiver pins and R_S (see Figure 7-22).

Use of ferrite beads on the D+ or D- lines of full-speed devices is discouraged.

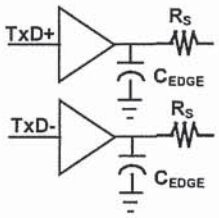


Figure 7-22. Placement of Optional Edge Rate Control Capacitors for Low-/full-speed

7.1.6.2 High-speed Input Characteristics

Figure 7-23 shows the simple equivalent loading circuit of a USB device operating in high-speed receive mode.

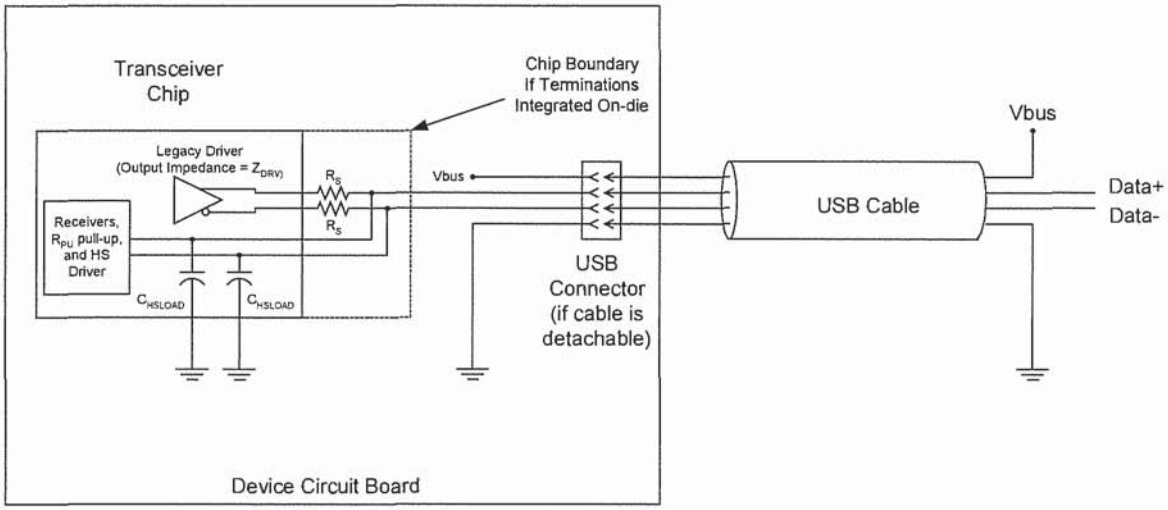


Figure 7-23. Diagram for High-speed Loading Equivalent Circuit

When operating in high-speed signaling mode, a transceiver must meet the following loading specifications:

1. DC output voltage and resistance specifications
2. TDR loading specification

Additionally, it is strongly recommended that a transceiver component operating in high-speed signaling mode should meet the following lumped capacitance guideline.

The use of ferrites on high-speed data lines is strongly discouraged.

DC output voltage and resistance specifications – A transceiver that is in high-speed mode must present a DC load on each of the data lines nominally equivalent to 45 Ω to ground. The actual resistance, Z_{HSDRV}, must be 40.5 Ω ≤ Z_{HSDRV} ≤ 49.5 Ω. The output voltage in the high-speed idle state (V_{HSTERM}) is specified in Table 7-3

TDR loading specification – The AC loading specifications of a transceiver in the high-speed idle state are specified in terms of differential TDR (Time Domain Reflectometer) measurements.

These measurements govern the maximum allowable transmission line discontinuities for the port connector, the interconnect leading from the connector to the transceiver, the transceiver package, and the transceiver IC itself. In the special case of a high-speed capable device with a captive cable, the transmission line discontinuities of the cable assembly are also governed.

The following specifications must be met with the incident rise time of the differential TDR set to 400 ps. It is important to note that all times are "as displayed" on the TDR and are hence "round trip times."

Termination Impedance (Z_{HSTERM}) is measured on the TDR trace at a specific measurement time following the connector reference time. The connector reference time is determined by disconnecting the TDR connection from the port connector and noting the time of the open circuit step. For an A connector, the measurement time is 8 ns after the connector reference location. For a B connector, the measurement time is 4 ns after the connector reference location. The differential termination impedance must be:

$$80\ \Omega \leq Z_{HSTERM} \leq 100\ \Omega$$

Through Impedance (Z_{HSTHRU}) is the impedance measured from 500 ps before the connector reference location until the time governed by the Termination impedance specification.

$$70\ \Omega \leq Z_{HSTHRU} \leq 110\ \Omega$$

In the Exception Window (a sliding 1.4 ns window inside the Through Impedance time window), the differential impedance may exceed the Through limits. No single excursion, however, may exceed the Through limits for more than twice the TDR rise time (400 ps).

In the special case of a high-speed capable device with a captive cable, the same specifications must be met, but the TDR measurements must be made through the captive cable assembly. Determination of the connector reference time can be more difficult in this case, since the cable may not be readily removable from the port being tested. It is left to the tester of a specific device to determine the connector reference location by whatever means are available.

Lumped capacitance guideline for the transceiver component

When characterizing a transceiver chip as an isolated component, the measurement can be performed effectively at the chip boundary shown in Figure 7-23 without USB connectors or cables. Parasitic capacitance of the test fixture can be corrected by measuring the capacitance of the fixture itself and subtracting this reading from the reading taken with the transceiver inserted. If the terminations are off-chip, discrete Rs resistors should be in place during the measurements, and measurements should be taken on the "connector side" of the resistors. The transceiver should be in Test_SE0_NAK mode during testing.

Capacitance measurements are taken from each of the data lines to ground while the other line is left open. The instrument used to perform this measurement must be able to determine the effective capacitance to ground in the presence of the parallel effective resistance to ground.

Capacitance to Ground on each line: $C_{HSLOAD} \leq 10$ pF
Matching of Capacitances to Ground: ≤ 1.0 pF

The guideline is to allow no more than 5.0 pF for the transceiver die itself and no more than an additional 5 pF for the package. The differential capacitance across the transceiver inputs should be no more than 5.0 pF

7.1.7 Signaling Levels

The following sections specify signaling levels for low-speed, full-speed, and high-speed operation.

7.1.7.1 Low-/Full-speed Signaling Levels

Table 7-2 summarizes the USB signaling levels. The source is required to drive the levels specified in the second column, and the target is required to identify the correct bus state when it sees the levels in the third column. (Target receivers can be more sensitive as long as they are within limits specified in the fourth column.)

Table 7-2. Low-/full-speed Signaling Levels

Bus State	Signaling Levels		
	At originating source connector (at end of bit time)	At final target connector	
		Required	Acceptable
Differential "1"	D+ > V_{OH} (min) and D- < V_{OL} (max)	(D+) - (D-) > 200 mV and D+ > V_{IH} (min)	(D+) - (D-) > 200 mV
Differential "0"	D- > V_{OH} (min) and D+ < V_{OL} (max)	(D-) - (D+) > 200 mV and D- > V_{IH} (min)	(D-) - (D+) > 200 mV
Single-ended 0 (SE0)	D+ and D- < V_{OL} (max)	D+ and D- < V_{IL} (max)	D+ and D- < V_{IH} (min)
Single-ended 1 (SE1)	D+ and D- > V_{OSE1}(min)	D+ and D- > V_{IL} (max)	
Data J state: Low-speed Full-speed	 Differential "0" Differential "1"	 Differential "0" Differential "1"	
Data K state: Low-speed Full-speed	 Differential "1" Differential "0"	 Differential "1" Differential "0"	
Idle state: Low-speed Full-speed	NA	 D- > V_{IHZ} (min) and D+ < V_{IL} (max) D+ > V_{IHZ} (min) and D- < V_{IL} (max)	 D- > V_{IHZ} (min) and D+ < V_{IH} (min) D+ > V_{IHZ} (min) and D- < V_{IH} (min)
Resume state	Data K state	Data K state	
Start-of-Packet (SOP)	Data lines switch from Idle to K state		
End-of-Packet (EOP)[4]	SE0 for approximately 2 bit times[1] followed by a J for 1 bit time[3]	SE0 for ≥ 1 bit time[2] followed by a J state for 1 bit time	SE0 for ≥ 1 bit time[2] followed by a J state
Disconnect (at downstream port)	NA	SE0 for ≥2.5 µs	
Connect (at downstream port)	NA	Idle for ≥2 ms	Idle for ≥2.5 µs
Reset	D+ and D- < VOL (max) for ≥10ms	D+ and D- < VIL (max) for ≥10 ms	D+ and D- < VIL (max) for ≥2.5 µs

Note 1: The width of EOP is defined in bit times relative to the speed of transmission. (Specification EOP widths are given in Table 7-7 and Table 7-8.)

Note 2: The width of EOP is defined in bit times relative to the device type receiving the EOP. The bit time is approximate.

Note 3: The width of the J state following the EOP is defined in bit times relative to the buffer edge rate. The J state from a low-speed buffer must be a low-speed bit time wide and, from a full-speed buffer, a full-speed bit time wide.

Note 4: The keep-alive is a low-speed EOP.

The J and K data states are the two logical levels used to communicate differential data in the system. Differential signaling is measured from the point where the data line signals cross over. Differential data signaling is not concerned with the level at which the signals cross, as long as the crossover voltage meets the requirements in Section 7.1.2. Note that, at the receiver, the Idle and Resume states are logically equivalent to the J and K states respectively.

As shown in Table 7-2, the J and K states for full-speed signaling are inverted from those for low-speed signaling. The sense of data, idle, and resume signaling is set by the type of device that is being attached to a port. If a full-speed device is attached to a port, that segment of the USB uses full-speed signaling conventions (and fast rise and fall times), even if the data being sent across the data lines is at the low-speed data rate. The low-speed signaling conventions shown in Table 7-2 (plus slow rise and fall times) are used only between a low-speed device and the port to which it is attached.

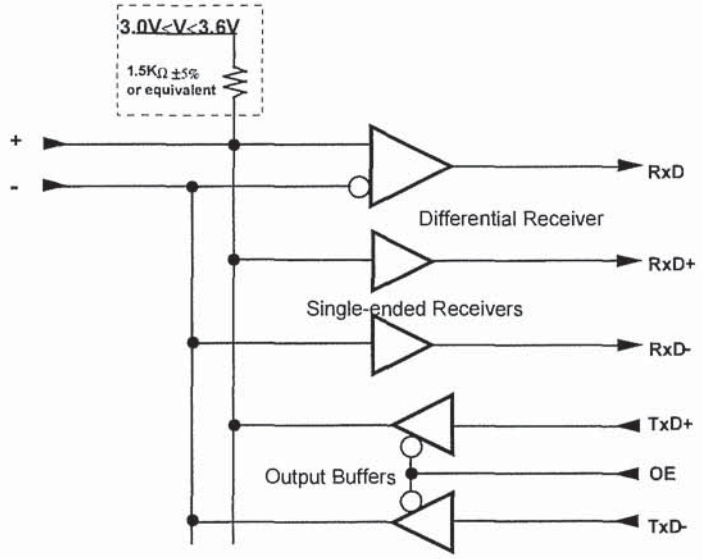


Figure 7-24. Upstream Facing Full-speed Port Transceiver

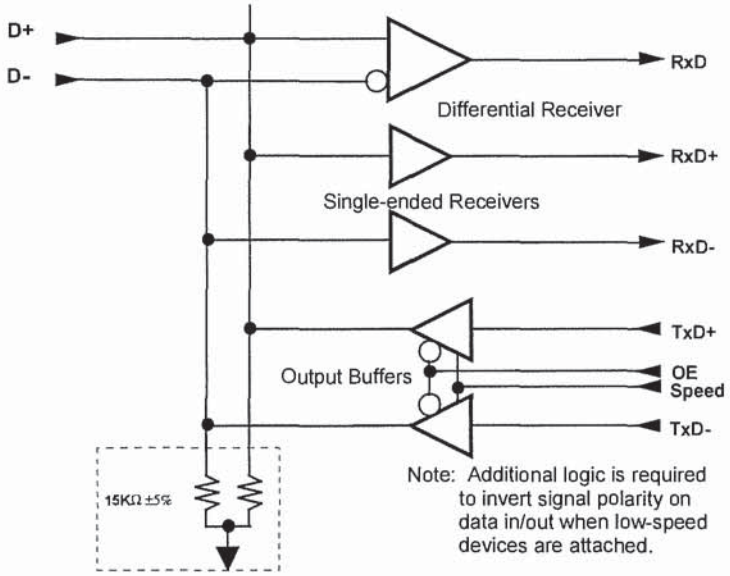


Figure 7-25. Downstream Facing Low-/full-speed Port Transceiver

7.1.7.2 Full-/High-speed Signaling Levels

The high-speed signaling voltage specifications in Table 7-3 must be met when measuring at the connector closest to the transceiver, using precision 45 Ω load resistors to the device ground as reference loads. All voltage measurements are taken with respect to the local device ground.

Table 7-3. High-speed Signaling Levels

Bus State	Required Signaling Level at Source Connector	Required Signaling Level at Target Connector
High-speed Differential "1"	**DC Levels:** V_{HSOH} (min) ≤ D+ ≤ V_{HSOH} (max) V_{HSOL} (min) ≤ D- ≤ V_{HSOL} (max) See Note 1. **AC Differential Levels:** A transmitter must conform to the eye pattern templates called out in Section 7.1.2. See Note 2.	**AC Differential Levels** The signal at the target connector must be recoverable, as defined by the eye pattern templates called out in Section 7.1.2. See Note 2.
High-speed Differential "0"	**DC Levels:** V_{HSOH} (min) ≤ D- ≤ V_{HSOH} (max) V_{HSOL} (min) ≤ D+ ≤ V_{HSOL} (max) See Note 1. **AC Differential Levels:** A transmitter must conform to the eye pattern templates called out in Section 7.1.2. See Note 2.	**AC Differential Levels:** The signal at the target connector must be recoverable, as defined by the eye pattern templates called out in Section 7.1.2. See Note 2.
High-speed J State	High-speed Differential "1"	High-speed Differential "1"
High-speed K State	High-speed Differential "0"	High-speed Differential "0"

Table 7-3. High-speed Signaling Levels (Continued)

Chirp J State (differential voltage; applies only during reset when both hub and device are high-speed capable)	**DC Levels:** V_{CHIRPJ} (min) ≤ (D+ - D-) ≤ V_{CHIRPJ} (max)	**AC Differential Levels** The differential signal at the target connector must be ≥ 300 mV
Chirp K State (differential voltage; applies only during reset when both hub and device are high-speed capable)	**DC Levels:** V_{CHIRPK} (min) ≤ (D+ - D-) ≤ V_{CHIRPK} (max)	**AC Differential Levels** The differential signal at the target connector must be ≤ -300 mV
High-speed Squelch State	NA	V_{HSSQ} - Receiver must indicate squelch when magnitude of differential voltage is ≤100 mV; receiver must not indicate squelch if magnitude of differential voltage is ≥150 mV. See Note 3.
High-speed Idle State	NA	**DC Levels:** V_{HSOI} min ≤ (D+, D-) ≤ V_{HSOI} max See Note 1. **AC Differential Levels:** Magnitude of differential voltage is ≤ 100 mV See Note 3.
Start of High-speed Packet (HSSOP)	Data lines switch from high-speed Idle to high-speed J or high-speed K state.	
End of High-speed Packet (HSEOP)	Data lines switch from high-speed J or K to high-speed Idle state.	
High-speed Disconnect State (at downstream facing port)	NA	V_{HSDSC} - Downstream facing port must not indicate device disconnection if differential voltage is ≤ 525 mV, and must indicate device disconnection when magnitude of differential voltage is ≥ 625 mV, at the sample time discussed in Section 7.1.7.3.

Note 1: Measured with a 45 Ω resistor to ground at each data line, using test modes Test_J and Test_K

Note 2: Measured using test mode Test_Packet with fixture shown in Figure 7-12

Note 3: Measured with fixture shown in Figure 7-12, using test mode SE0_NACK

Note 4: A high-speed driver must never "intentionally" generate a signal in which both D+ and D- are driven to a level above 200 mV. The current-steering design of a high-speed driver should naturally preclude this possibility.

7.1.7.3 Connect and Disconnect Signaling

When no function is attached to the downstream facing port of a host or hub in low-/full-speed, the pull-down resistors present there will cause both D+ and D- to be pulled below the single-ended low threshold of the host or hub transceiver when that port is not being driven by the hub. This creates an SE0 state on the downstream facing port. A disconnect condition is indicated if the host or hub is not driving the data lines and an SE0 persists on a downstream facing port for more than TDDIS (see Figure 7-26). The specifications for TDDIS and TDCNN are defined in Table 7-13.

A connect condition will be detected when the hub detects that one of the data lines is pulled above its VIH threshold for more than TDCNN (see Figure 7-27 and Figure 7-28).

Hubs must determine the speed of the attached device by sampling the state of the bus immediately before driving SE0 to indicate a reset condition to the device.

All signaling levels given in Table 7-2 are set for this bus segment (and this segment alone) once the speed of the attached device is determined. The mechanics of speed detection are described in Section 11.8.2.

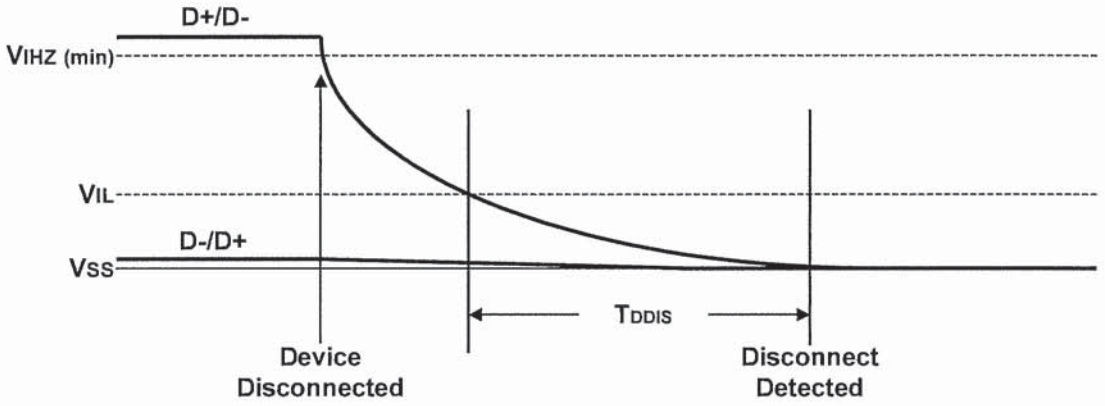


Figure 7-26. Low-/full-speed Disconnect Detection

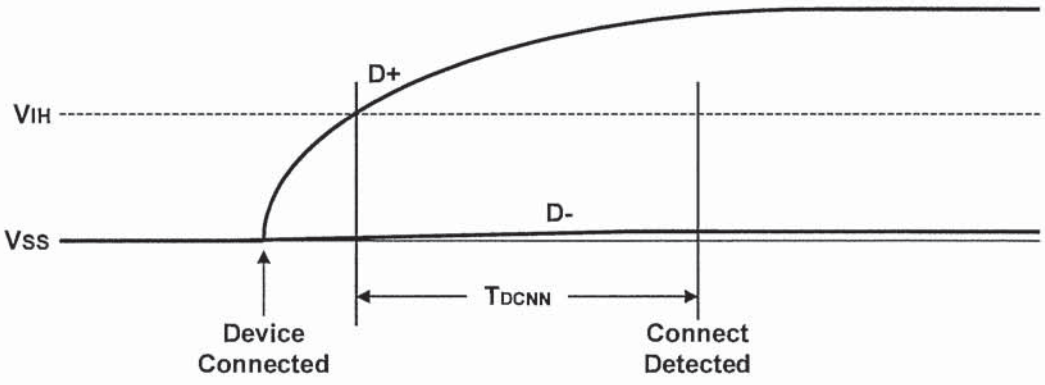


Figure 7-27. Full-/high-speed Device Connect Detection

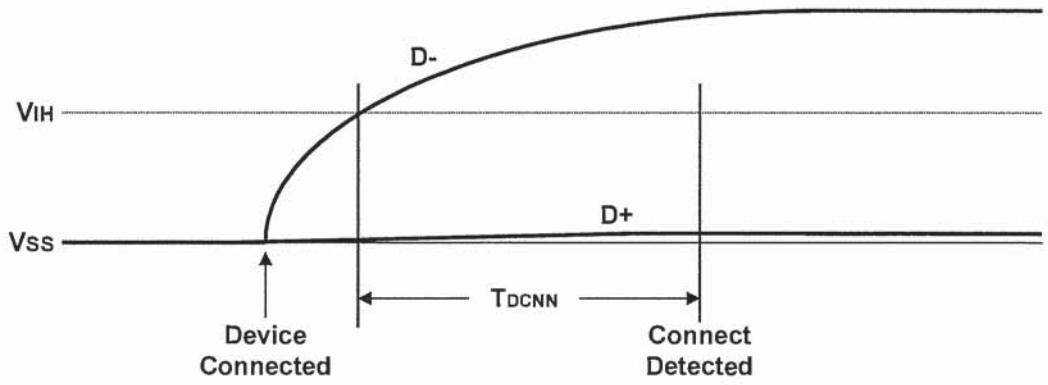


Figure 7-28. Low-speed Device Connect Detection

Because USB components may be hot plugged, and hubs may implement power switching, it is necessary to comprehend the delays between power switching and/or device attach and when the device's internal power has stabilized. Figure 7-29 shows all the events associated with both turning on port power with a device connected and hot-plugging a device. There are six delays and a sequence of events that are defined by this specification.

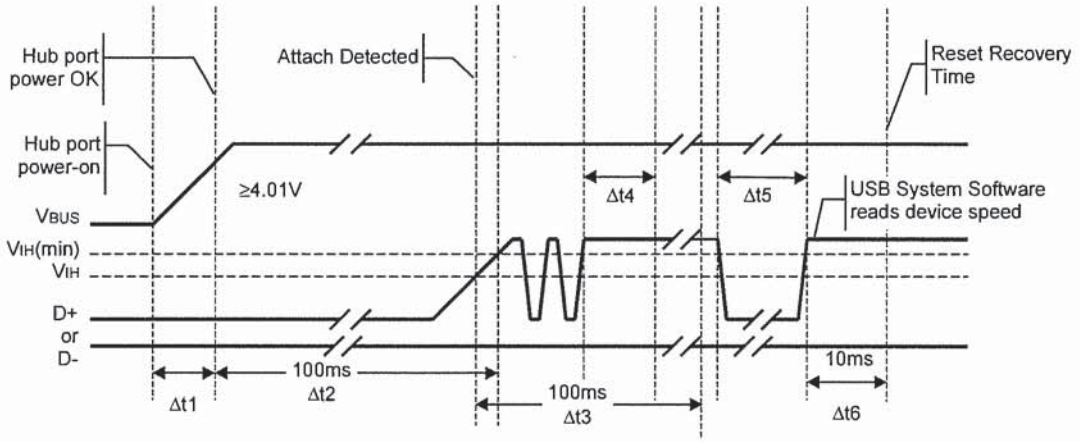


Figure 7-29. Power-on and Connection Events Timing

Δt1 This is the amount of time required for the hub port power switch to operate. This delay is a function of the type of hub port switch. Hubs report this time in the hub descriptor (see Section 11.15.2.1), which can be read via a request to the Hub Controller (see Section 11.16.2.4). If a device were plugged into a non-switched or already-switched on port, Δt1 is equal to zero.

Δt2 (TSIGATT) This is the maximum time from when VBUS is up to valid level (4.01 V) to when a device has to signal attach. Δt2 represents the time required for the device's internal power rail to stabilize and for D+ or D- to reach VIH (min) at the hub. Δt2 must be less than 100 ms for all hub and device implementations. (This requirement only applies if the device is drawing power from the bus.)

Δt3 (TATTDB) This is a debounce interval with a minimum duration of 100 ms that is provided by the USB System Software. It ensures that the electrical and mechanical connection is stable before software attempts to reset the attached device. The interval starts when the USB System Software is notified of a connection detection. The interval restarts if there is a disconnect. The debounce interval ensures that power is stable at the device for at least 100 ms before any requests will be sent to the device.

Δt4 (T2SUSP) Anytime a device observes no bus activity, it must obey the rules of going into suspend (see Section 7.1.7.6).

Δt5 (TDRST) This is the period of time hubs drive reset to a device. Refer to Section 7.1.7.5 and Section 11.5.1.5 for details.

Δt6 (TRSTRCY) The USB System Software guarantees a minimum of 10 ms for reset recovery. Device response to any bus transactions addressed to the default device address during the reset recovery time is undefined.

High-speed capable devices must initially attach as full-speed devices and must comply with all full-speed connection requirements. A high-speed capable downstream facing port must correctly detect the attachment of low-speed and full-speed devices and must also comply with all low-speed and full-speed connection behaviors.

Transition to high-speed signaling is accomplished by means of a low level electrical protocol which occurs during Reset. This protocol is specified in Section 7.1.7.5.

A downstream facing transceiver operating in high-speed mode detects disconnection of a high-speed device by sensing the doubling in differential signal amplitude across the D+ and D- lines that can occur when the device terminations are removed. The Disconnection Envelope Detector output goes high when the downstream facing transceiver transmits and positive reflections from the open line return with a phase which is additive with the transceiver driver signal. Signals with differential amplitudes ≥ 625 mV must reliably activate the Disconnection Envelope Detector. Signals with differential amplitudes ≤ 525 mV must never activate the Disconnection Envelope Detector.

To assure that this additive effect occurs and is of sufficient duration to be detected, the EOP at the end of a high-speed SOF is lengthened to a continuous string of 40 bits without any transitions, as discussed in Section 7.1.13.2. This length is sufficient to guarantee that the voltage at the downstream facing port's connector will double, since the maximum allowable round trip signal delay is 30 bit times.

When a downstream facing port is transmitting in high-speed mode and detects that it has sent 32 bits without a transition, the disconnection envelope detector's output must be sampled once during transmission of the next 8 bits at the transceiver output. (In the absence of bus errors, the next 8 bits will not include a transition.) If the sample indicates that the disconnection detection threshold has been exceeded, the downstream facing port must indicate that the high-speed device has been disconnected. See Section 11.12.4.

7.1.7.4 Data Signaling

Data transmission within a packet is done with differential signals.

7.1.7.4.1 Low-/Full-Speed Signaling

The start of a packet (SOP) is signaled by the originating port by driving the D+ and D- lines from the Idle state to the opposite logic level (K state). This switch in levels represents the first bit of the SYNC field. Hubs must limit the change in the width of the first bit of SOP when it is retransmitted to less than ± 5 ns. Distortion can be minimized by matching the nominal data delay through the hub with the output enable delay of the hub.

The SE0 state is used to signal an end-of-packet (EOP). EOP will be signaled by driving D+ and D- to the SE0 state for two bit times followed by driving the lines to the J state for one bit time. The transition from the SE0 to the J state defines the end of the packet at the receiver. The J state is asserted for one bit time and then both the D+ and D- output drivers are placed in their high-impedance state. The bus termination resistors hold the bus in the Idle state. Figure 7-30 shows the signaling for start and end of a packet.

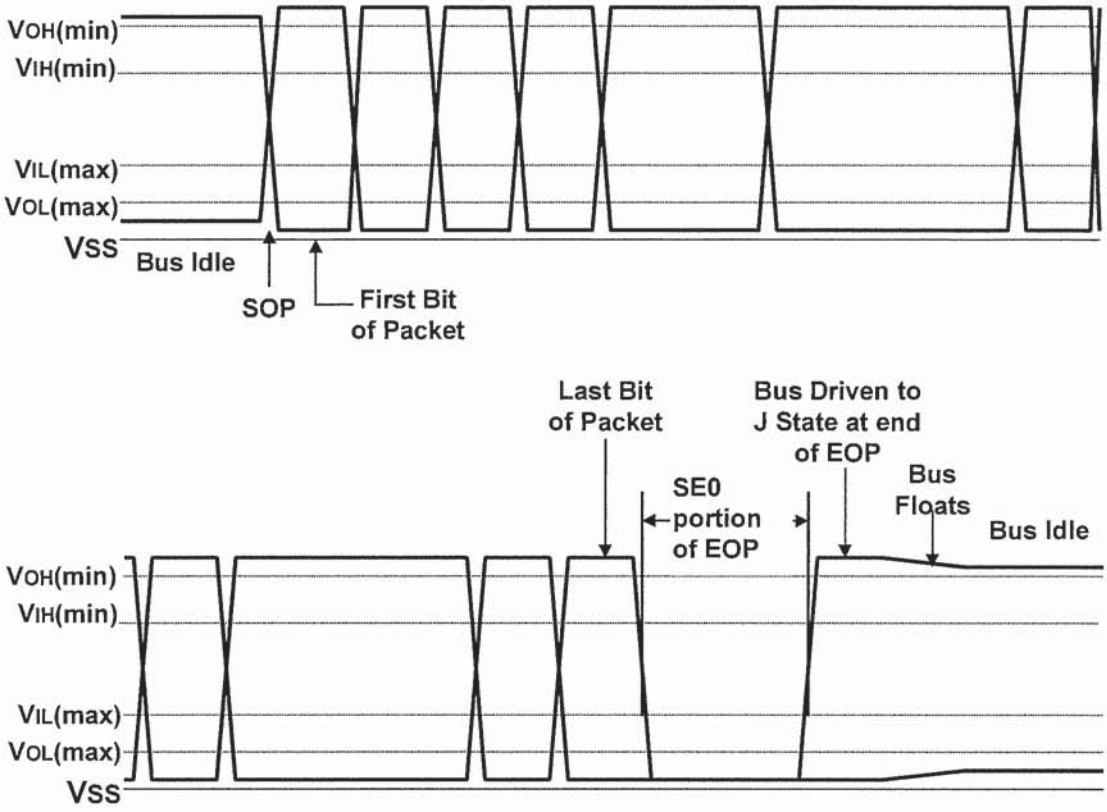


Figure 7-30. Low-/full-speed Packet Voltage Levels

7.1.7.4.2 High-speed Signaling

The high-speed Idle state is when both lines are nominally at GND.

The source of the packet signals the Start of Packet (SOP) in high-speed mode by driving the D+ and D- lines from the high-speed Idle state to the K state. This K is the first symbol of the SYNC pattern (NRZI sequence KJKJKJKJ KJKJKJKJ KJKJKJKJ KJKJKJKK) as described in Section 7.1.10.

The high-speed End of Packet (EOP) begins with a transition from the last symbol before the EOP to the opposite symbol. This opposite symbol is the first symbol in the EOP pattern (NRZ 01111111 with bit stuffing disabled) as described in Section 7.1.13.2. Upon completion of the EOP pattern, the driver ceases to inject current into the D+ or D- lines, and the lines return to the high-speed Idle state. The high-speed SOF EOP is a special case. This SOF EOP is 40 symbols without a transition (rather than 8 for a non-SOF packet).

The fact that the first symbol in the EOP pattern forces a transition simplifies the process of determining precisely which is the last bit in the packet prior to the EOP delimiter.

7.1.7.5 Reset Signaling

A hub signals reset to a downstream port by driving an extended SE0 at the port. After the reset is removed, the device will be in the Default state (refer to Section 9.1).

The reset signaling can be generated on any Hub or Host Controller port by request from the USB System Software. The reset signaling must be driven for a minimum of 10ms (TDRST). After the reset, the hub port will transition to the Enabled state (refer to Section 11.5).

As an additional requirement, Host Controllers and the USB System Software must ensure that resets issued to the root ports drive reset long enough to overwhelm any concurrent resume attempts by downstream devices. It is required that resets from root ports have a duration of at least 50 ms (TDRSTR). It is not required that this be 50 ms of continuous Reset signaling. However, if the reset is not continuous, the interval(s) between reset signaling must be less than 3 ms (TRHRSI), and the duration of each SE0 assertion must be at least 10 ms (TDRST).

A device operating in low-/full-speed mode that sees an SE0 on its upstream facing port for more than 2.5 µs (TDETRST) may treat that signal as a reset. The reset must have taken effect before the reset signaling ends.

Hubs will propagate traffic to a newly reset port after the port is in the Enabled state. The device attached to this port must recognize this bus activity and keep from going into the Suspend state.

Hubs must be able to accept all hub requests and devices must be able to accept a SetAddress() request (refer to Section 11.24.2 and Section 9.4 respectively) after the reset recovery time 10 ms (TRSTRCY) after the reset is removed. Failure to accept this request may cause the device not to be recognized by the USB system software. Hubs and devices must complete commands within the times specified in Chapter 9 and Chapter 11.

Reset must wake a device from the Suspend state.

It is required that a high-speed capable device can be reset while in the Powered, Default, Address, Configured, or Suspended states shown in Figure 9-1. The reset signaling is compatible with low-/full-speed reset. This means that a hub must successfully reset any device (even USB 1.X devices), and a device must be successfully reset by any hub (even USB1.X hubs).

If, and only if, a high-speed capable device is reset by a high-speed capable hub which is not high-speed disallowed, both hub and device must be operating in the default state in high-speed signaling mode at the end of reset. The hub port status register must indicate that the port is in high-speed signaling mode. This requirement is met by having such a device and such a hub engage in a low level protocol during the reset signaling time. The protocol is defined in such a way that USB 1.X devices will not be disrupted from their normal reset behaviors.

Note: Because the downstream facing port will not be in Transmit state during the Reset Protocol, high-speed Chirp signaling levels will not provoke disconnect detection. (Refer to Section 7.1.7.3 and Section 11.5.1.7.)

Reset Protocol for high-speed capable hubs and devices

1. The hub checks to make sure the attached device is not low-speed. (A low-speed device is not allowed to support high-speed operation. If the hub determines that it is attached to a low-speed device, it does not conduct the following high-speed detection protocol during reset.)

2. The hub drives SE0. In this description of the Reset Protocol and High-speed Detection Handshake, the start of SE0 is referred to as time T0.

3. The device detects assertion of SE0.

   a) If the device is being reset from suspend, then the device begins a high-speed detection handshake after the detection of SE0 for no less than 2.5 µs (TFILTSE0). Since a suspended device will generally have its clock oscillator disabled, the detection of SE0 will cause the oscillator to be restarted. The clock must be useable (although not necessarily settled to 500 ppm accuracy) in time to detect the high-speed hub chirp as described in Step 8.
   b) If the device is being reset from a non-suspended full-speed state, then the device begins a high-speed detection handshake after the detection of SE0 for no less than 2.5 µs and no more than 3.0 ms (TWTRSTFS).
   c) If the device is being reset from a non-suspended high-speed state, then the device must wait no less than 3.0 ms and no more than 3.125 ms (TWTREV) before reverting to full-speed. Reversion to full-speed is accomplished by removing the high-speed termination and reconnecting the D+ pull-up resistor. The device samples the bus state, and checks for SE0 (reset as opposed to suspend), no less than 100 µs and no more than 875 µs (TWTRSTHS) after starting reversion to full-speed. If SE0 (reset) is detected, then the device begins a high-speed detection handshake.

High-speed Detection Handshake (not performed if low-speed device detected by hub):

Note: In the following handshake, both the hub and device are required to detect Chirp J's and K's of specified minimum durations. It is strongly recommended that "gaps" in these Chirp signals as short as 16 high-speed bit times should restart the duration timers.

4. The high-speed device leaves the D+ pull-up resistor connected, leaves the high-speed terminations disabled, and drives the high-speed signaling current into the D- line. This creates a Chirp K on the bus. The device chirp must last no less than 1.0 ms (TUCH) and must end no more than 7.0 ms (TUCHEND) after high-speed Reset time T0.

5. The hub must detect the device chirp after it has seen assertion of the Chirp K for no less than 2.5 µs (TFILT). If the hub does not detect a device chirp, it must continue the assertion of SE0 until the end of reset.

6. No more than 100 µs (TWTDCH) after the bus leaves the Chirp K state, the hub must begin to send an alternating sequence of Chirp K's and Chirp J's. There must be no Idle states on the bus between the J's and K's. This sequence must continue until a time (TDCHSE0) no more than 500 µs before and no less than 100 µs before the end of Reset. (This will guarantee that the bus remains active, preventing the device from entering the high-speed Suspend state.) Each individual Chirp K and Chirp J must last no less than 40 µs and no more than 60 µs (TDCHBIT).

7. After completing the hub chirp sequence, the hub asserts SE0 until end of Reset. At the end of reset, the hub must transition to the high-speed Enabled state without causing any transitions on the data lines.

8. After the device completes its chirp, it looks for the high-speed hub chirp. At a minimum, the device is required to see the sequence Chirp K-J-K-J-K-J in order to detect a valid hub chirp. Each individual Chirp K and Chirp J must be detected for no less than 2.5 µs (TFILT).

   a) If the device detects the sequence Chirp K-J-K-J-K-J, then no more than 500 µs (TWTHS) after detection, the device is required to disconnect the D+ pull-up resistor, enable the high-speed terminations, and enter the high-speed Default state.
   b) If the device has not detected the sequence Chirp K-J-K-J-K-J by a time no less than 1.0 ms and no more than 2.5 ms (TWTFS) after completing its own chirp, then the device is required to revert to the full-speed Default state and wait for the end of Reset.

7.1.7.6 Suspending

All devices must support the Suspend state. Devices can go into the Suspend state from any powered state. They begin the transition to the Suspend state after they see a constant Idle state on their upstream facing bus lines for more than 3.0 ms. The device must actually be suspended, drawing only suspend current from the bus after no more than 10 ms of bus inactivity on all its ports. Any bus activity on the upstream facing port will keep

a device out of the Suspend state. In the absence of any other bus traffic, the SOF token (refer to Section 8.4.3) will occur once per (micro)frame to keep full-/high-speed devices from suspending. In the absence of any low-speed traffic, low-speed devices will see at least one keep-alive (defined in Table 7-2) in every frame in which an SOF occurs, which keeps them from suspending. Hubs generate this keep-alive as described in Section 11.8.4.1.

While in the Suspend state, a device must continue to provide power to its D+ (full-/high-speed) or D- (low-speed) pull-up resistor to maintain an idle so that the upstream hub can maintain the correct connectivity status for the device.

Additional Requirements for High-speed Capable Devices

From the perspective of a device operating in high-speed mode, a Reset and a Suspend are initially indistinguishable, so the first part of the device response is the same as for a Reset. When a device operating in high-speed mode detects that the data lines have been in the high-speed Idle state for at least 3.0 ms, it must revert to the full-speed configuration no later than 3.125 ms (TWTREV) after the start of the idle state. Reversion to full-speed is accomplished by disconnecting its termination resistors and reconnecting its D+ pull-up resistor. No earlier than 100 µs and no later than 875 µs (TWTRSTHS) after reverting to full-speed, the device must sample the state of the line. If the state is a full-speed J, the device continues with the Suspend process. (SE0 would have indicated that the downstream facing port was driving reset, and the device would have gone into the "High-speed Detection Handshake" as described in Section 7.1.7.5.)

A device or downstream facing port which is suspended from high-speed operation actually transitions to full-speed signaling during the suspend process, but is required to remember that it was operating in high-speed mode when suspended. When the resume occurs, the device or downstream facing transceiver must revert to high-speed as discussed in Section 7.1.7.7 without the need for a reset.

7.1.7.6.1 Global Suspend

Global suspend is used when no communication is desired anywhere on the bus and the entire bus is placed in the Suspend state. The host signals the start of global suspend by ceasing all its transmissions (including the SOF token). As each device on the bus recognizes that the bus is in the Idle state for the appropriate length of time, it goes into the Suspend state.

After 3.0 ms of continuous idle state, a downstream facing transceiver operating in high-speed must revert to the full-speed idle configuration (high-speed terminations disabled), but it does not enable full-speed disconnect detection until 1.0 ms later. This is to make sure that the device has returned to the full-speed Idle state prior to the enabling of full-speed disconnect detection, thereby preventing an unintended disconnect detection. After re-enabling the full-speed disconnect detection mechanism, the hub continues with the suspend process.

7.1.7.6.2 Selective Suspend

Segments of the bus can be selectively suspended by sending the command SetPortFeature(PORT_SUSPEND) to the hub port to which that segment is attached. The suspended port will block activity to the suspended bus segment, and devices on that segment will go into the Suspend state after the appropriate delay as described above.

When a downstream facing port operating in high-speed mode receives the SetPortFeature(PORT_SUSPEND) command, the port immediately reverts to the full-speed Idle state and blocks any activity to the suspend segment. Full-speed disconnect detection is disabled until the port has been in full-speed idle for 4.0 ms. This prevents an unintended disconnect detection. After re-enabling the full-speed disconnect detection mechanism, the hub continues with the suspend process.

Section 11.5 describes the port Suspend state and its interaction with the port state machine. Suspend is further described in Section 11.9.

7.1.7.7 Resume

If a device is in the Suspend state, its operation is resumed when any non-idle signaling is received on its upstream facing port. Additionally, the device can signal the system to resume operation if its remote wakeup capability has been enabled by the USB System Software. Resume signaling is used by the host or a device to bring a suspended bus segment back to the active condition. Hubs play an important role in the propagation and generation of resume signaling. The following description is an outline of a general global resume sequence. A complete description of the resume sequence, the special cases caused by selective suspend, and the role of the hub are given in Section 11.9.

The host may signal resume (TDRSMDN) at any time. It must send the resume signaling for at least 20 ms and then end the resume signaling in one of two ways, depending on the speed at which its port was operating when it was suspended. If the port was in low-/full-speed when suspended, the resume signaling must be ended with a standard, low-speed EOP (two low-speed bit times of SE0 followed by a J). If the port was operating in high-speed when it was suspended, the resume signaling must be ended with a transition to the high-speed idle state.

The 20 ms of resume signaling ensures that all devices in the network that are enabled to see the resume are awakened. The connectivity established by the resume signaling is torn down by the End of Resume, which prepares the hubs for normal operation. After resuming the bus, the host must begin sending bus traffic (at least the SOF token) within 3 ms of the start of the idle state to keep the system from going back into the Suspend state.

A device with remote wakeup capability may not generate resume signaling unless the bus has been continuously in the Idle state for 5 ms (TWTRSM). This allows the hubs to get into their Suspend state and prepare for propagating resume signaling. The remote wakeup device must hold the resume signaling for at least 1 ms but for no more than 15 ms (TDRSMUP). At the end of this period, the device stops driving the bus (puts its drivers into the high-impedance state and does not drive the bus to the J state).

If the hub upstream of a remote wakeup device is suspended, it will propagate the resume signaling to its upstream facing port and to all of its enabled downstream facing ports, including the port that originally signaled the resume. When a hub is propagating resume signaling from a downstream device, it may transition from the idle state to K with a risetime faster than is normally allowed. The hub must begin this rebroadcast (TURSM) of the resume signaling within 1 ms of receiving the original resume. The resume signal will propagate in this manner upstream until it reaches the host or a non-suspended hub (refer to Section 11.9), which will reflect the resume downstream and take control of resume timing. This hub is termed the controlling hub. Intermediate hubs (hubs between the resume initiator and the controlling hub) drive resume (TDRSMUP) on their upstream facing port for at least 1 ms during which time they also continue to drive resume on enabled downstream facing ports. An intermediate hub will stop driving resume on the upstream facing port and reverse the direction of connectivity from upstream to downstream within 15 ms after first asserting resume on its upstream facing port. When all intermediate hubs have reversed connectivity, resume is being driven from the controlling hub through all intermediate hubs and to all enabled ports. The controlling hub must rebroadcast the resume signaling within 1 ms (TURSM) and ensures that resume is signaled for at least 20 ms (TDRSMDN). The hub may then begin normal operation by terminating the resume process as described above.

The USB System Software must provide a 10 ms resume recovery time (TRSMRCY) during which it will not attempt to access any device connected to the affected (just-activated) bus segment.

Port connects and disconnects can also cause a hub to send a resume signal and awaken the system. These events will cause a hub to send a resume signal only if the hub has been enabled as a remote-wakeup source. Refer to Section 11.4.4 for more details.

Refer to Section 7.2.3 for a description of power control during suspend and resume.

If the hub port and device were operating in high-speed prior to suspend, they are required to "remember" that they were previously operating in high-speed, and they must transition back to high-speed operation, without arbitration, within two low-speed bit times of the K to SE0 transition. The inactivity timers must be started two low-speed bit times after the K to SE0 transition. Note that the transition from SE0 to J which would normally

occur at the end of full-speed resume signaling is omitted if the link was operating in high-speed at the time when it was suspended.

It is required that the host begin sending SOF's in time to prevent the high-speed tree from suspending.

7.1.8 Data Encoding/Decoding

The USB employs NRZI data encoding when transmitting packets. In NRZI encoding, a "1" is represented by no change in level and a "0" is represented by a change in level. Figure 7-31 shows a data stream and the NRZI equivalent. The high level represents the J state on the data lines in this and subsequent figures showing NRZI encoding. A string of zeros causes the NRZI data to toggle each bit time. A string of ones causes long periods with no transitions in the data.

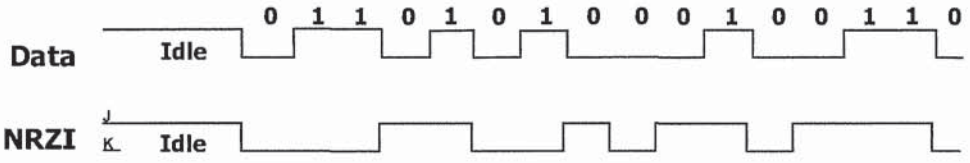


Figure 7-31. NRZI Data Encoding

7.1.9 Bit Stuffing

In order to ensure adequate signal transitions, bit stuffing is employed by the transmitting device when sending a packet on USB (see Figure 7-32 and Figure 7-34). A zero is inserted after every six consecutive ones in the data stream before the data is NRZI encoded, to force a transition in the NRZI data stream. This gives the receiver logic a data transition at least once every seven bit times to guarantee the data and clock lock. Bit stuffing is enabled beginning with the Sync Pattern. The data "one" that ends the Sync Pattern is counted as the first one in a sequence. Bit stuffing by the transmitter is always enforced, except during high-speed EOP. If required by the bit stuffing rules, a zero bit will be inserted even if it is the last bit before the end-of-packet (EOP) signal.

The receiver must decode the NRZI data, recognize the stuffed bits, and discard them.

7.1.9.1 Full-/low-speed

Full-/low-speed signaling uses bit stuffing throughout the packet without exception. If the receiver sees seven consecutive ones anywhere in the packet, then a bit stuffing error has occurred and the packet should be ignored. The time interval just before an EOP is a special case. The last data bit before the EOP can become stretched by hub switching skews. This is known as dribble and can lead to the case illustrated in Figure 7-33, which shows where dribble introduces a sixth bit that does not require a bit stuff. Therefore, the receiver must accept a packet for which there are up to six full bit times at the port with no transitions prior to the EOP.

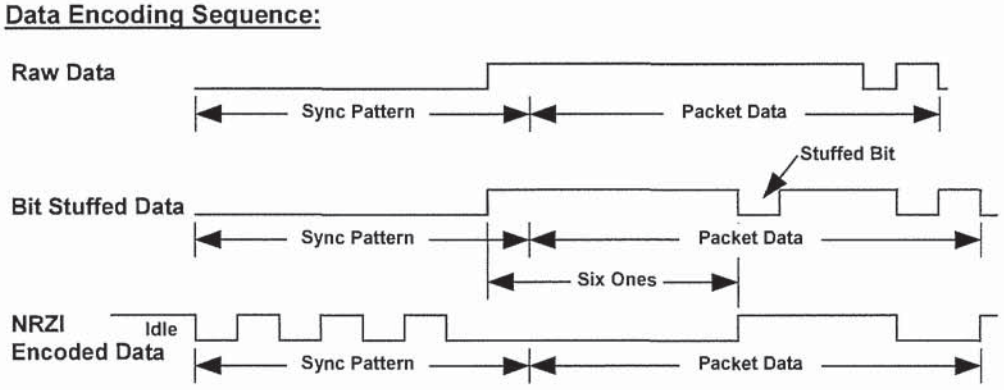


Figure 7-32. Bit Stuffing

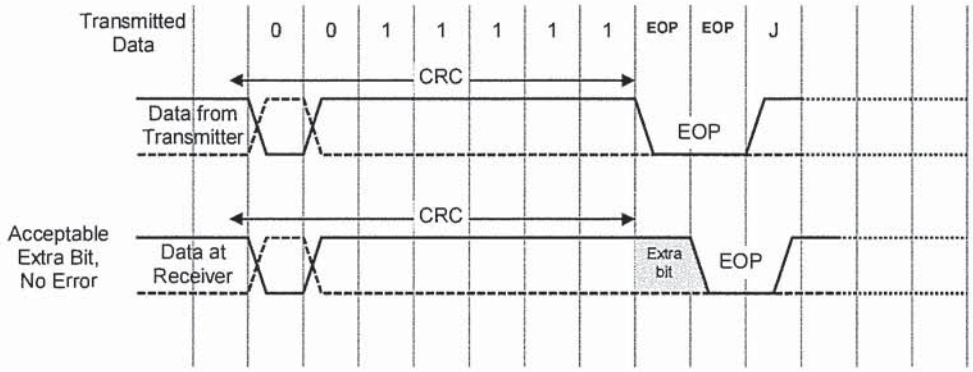


Figure 7-33. Illustration of Extra Bit Preceding EOP (Full-/low-speed)

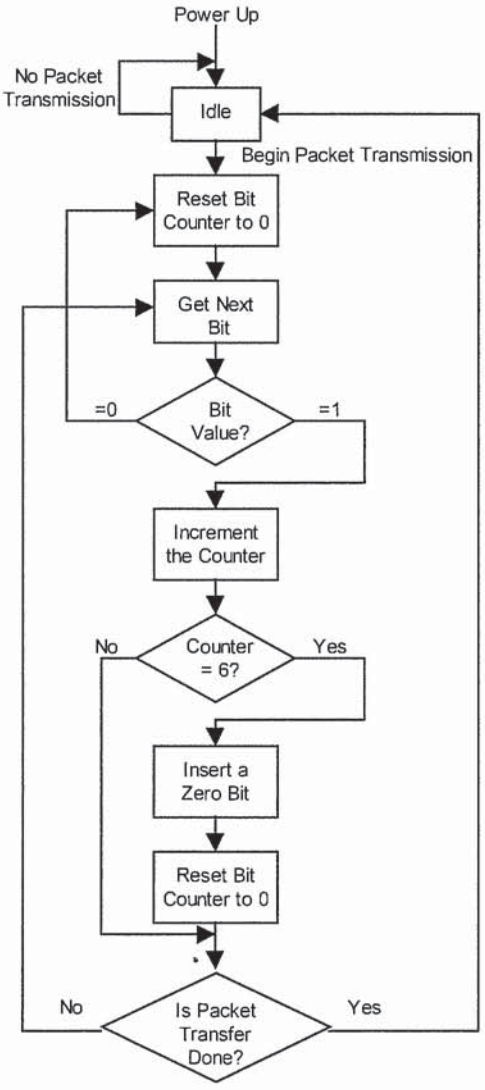


Figure 7-34. Flow Diagram for Bit Stuffing

7.1.9.2 High-Speed

High-speed signaling uses bit stuffing throughout the packet, with the exception of the intentional bit stuff errors used in the high-speed EOP as described in Section 7.1.13.2.

7.1.10 Sync Pattern

The SYNC pattern used for low-/full-speed transmission is required to be 3 KJ pairs followed by 2 K's for a total of eight symbols. Figure 7-35 shows the NRZI bit pattern, which is prefixed to each low-/full-speed packet.

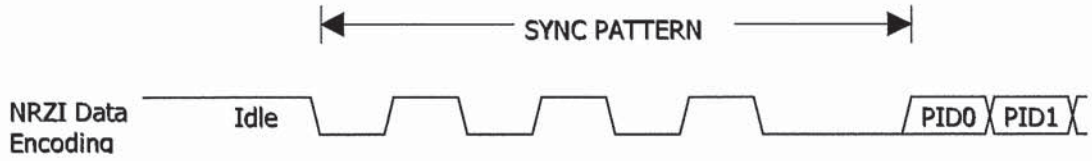


Figure 7-35. Sync Pattern (Low-/full-speed)

The SYNC pattern used for high-speed transmission is required to be 15 KJ pairs followed by 2 K's, for a total of 32 symbols. Hubs are allowed to drop up to 4 bits from the start of the SYNC pattern when repeating packets. Hubs must not corrupt any repeated bits of the SYNC field, however. Thus, after being repeated by 5 hubs, a packet's SYNC field may be as short as 12 bits.

7.1.11 Data Signaling Rate

The high-speed data rate (THSDRAT) is nominally 480.00 Mb/s, with a required bit rate accuracy of ±500 ppm. For hosts, hubs, and high-speed capable functions, the required data-rate accuracy when transmitting at any speed is ±0.05% (500 ppm). The full-speed rate for such hubs and functions is TFDRATHS. The low-speed rate for such hubs is TLDRATHS (a low-speed function must not support high-speed).

The full-speed data rate is nominally 12.000 Mb/s. For full-speed only functions, the required data-rate when transmitting (TFDRATE) is 12.000 Mb/s ±0.25% (2,500 ppm).

The low-speed data rate is nominally 1.50 Mb/s. For low-speed functions, the required data-rate when transmitting (TLDRATE) is 1.50 Mb/s ±1.5% (15,000 ppm). This allows the use of resonators in low cost, low-speed devices.

Hosts and hubs must be able to receive data from any compliant low-speed, full-speed, or high-speed source. High-speed capable functions must be able to receive data from any compliant full-speed or high-speed source. Full-speed only functions must be able to receive data from any compliant full-speed source. Low-speed only functions must be able to receive data from any compliant low-speed source.

The above accuracy numbers include contributions from all sources:

- Initial frequency accuracy
- Crystal capacitive loading
- Supply voltage on the oscillator
- Temperature
- Aging

7.1.12 Frame Interval

The USB defines a frame interval (TFRAME) to be 1.000 ms ±500 ns long. The USB defines a microframe interval (THSFRAM) to be 125.0 µs ±62.5 ns long. The (micro)frame interval is measured from any point in an SOF token in one (micro)frame to the same point in the SOF token of the next (micro)frame.

Since the Host Controller and hubs must meet clock accuracy specification of ±0.05%, they will automatically meet the frame interval requirements without the need for adjustment.

The frame interval repeatability, TRFI (difference in frame interval between two successive frames), must be less than 0.5 full-speed bit times. The microframe interval repeatability, THSRFI (difference in the microframe interval between two successive microframes, measured at the host), must be less than 4 high-speed bit times. Each hub may introduce at most 4 additional high-speed bits of microframe jitter.

Hubs and certain full-/high-speed functions need to track the (micro)frame interval. They also are required to have sufficient frame timing adjustment to compensate for their own frequency inaccuracy.

7.1.13 Data Source Signaling

This section covers the timing characteristics of data produced and sent from a port (the data source). Section 7.1.14 covers the timing characteristics of data that is transmitted through the Hub Repeater section of a hub. In this section, TPERIOD is defined as the actual period of the data rate that can have a range as defined in Section 7.1.11.

7.1.13.1 Data Source Jitter

This section describes the maximum allowable data source jitter for low-speed, full-speed, and high-speed signaling.

7.1.13.1.1 Low-/full-speed Data Source Jitter

The source of data can have some variation (jitter) in the timing of edges of the data transmitted. The time between any set of data transitions is (N * TPERIOD) ± jitter time, where 'N' is the number of bits between the transitions. The data jitter is measured with the same load used for maximum rise and fall times and is measured at the crossover points of the data lines, as shown in Figure 7-36.

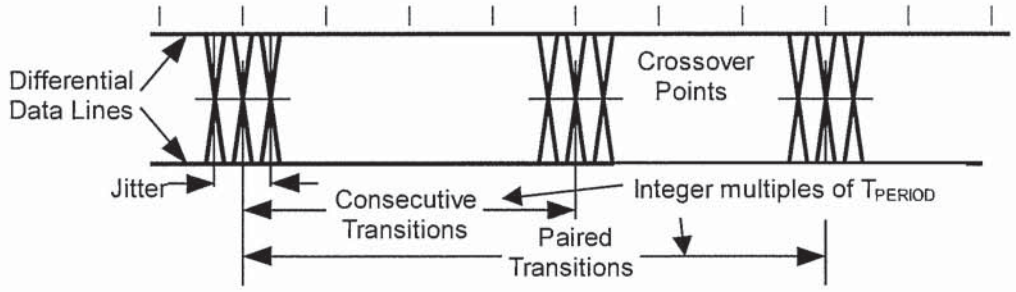


Figure 7-36. Data Jitter Taxonomy

- For full-speed transmissions, the jitter time for any consecutive differential data transitions must be within ±2.0 ns and within ±1.0 ns for any set of paired (JK-to-next JK transition or KJ-to-next KJ transition) differential data transitions.
- For low-speed transmissions, the jitter time for any consecutive differential data transitions must be within ±25 ns and within ±10 ns for any set of paired differential data transitions.

These jitter numbers include timing variations due to differential buffer delay and rise and fall time mismatches, internal clock source jitter, and noise and other random effects.

7.1.13.1.2 High-speed Data Source Jitter

High-speed data within a single packet must be transmitted with no more jitter than is allowed by the eye patterns defined in Section 7.1.2 when measured over a sliding window of 480 high-speed bit times.

7.1.13.2 EOP Width

This section describes low-speed, full-speed, and high-speed EOP width.

7.1.13.2.1 Low-/full-speed EOP

The width of the SE0 in the EOP is approximately 2 * TPERIOD. The SE0 width is measured with the same load used for maximum rise and fall times and is measured at the same level as the differential signal crossover points of the data lines (see Figure 7-37).

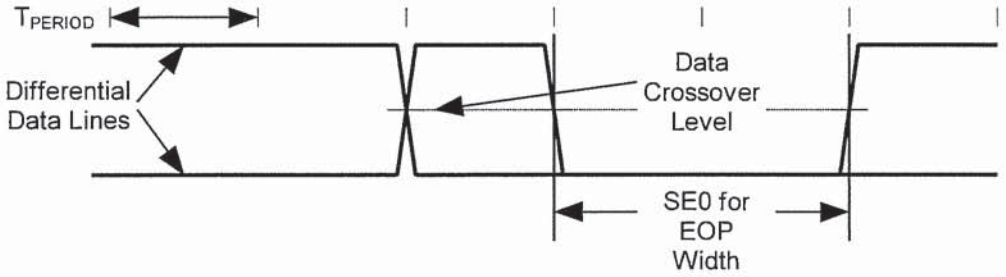


Figure 7-37. SE0 for EOP Width Timing

- For full-speed transmissions, the SE0 for EOP width from the transmitter must be between 160 ns and 175 ns.
- For low-speed transmissions, the transmitter's SE0 for EOP width must be between 1.25 µs and 1.50 µs.

These ranges include timing variations due to differential buffer delay and rise and fall time mismatches and to noise and other random effects.

A receiver must accept any valid EOP. Receiver design should note that the single-ended input threshold voltage can be different from the differential crossover voltage and the SE0 transitions will in general be asynchronous to the clock encoded in the NRZI stream.

- A full-speed EOP may have the SE0 interval reduced to as little as 82 ns (TFEOPR) and a low-speed SE0 interval may be as short as 670 ns (TLEOPR).

A hub may tear down connectivity if it sees an SE0 of at least TFST or TLST followed by a transition to the J state. A hub must tear down connectivity on any valid EOP.

7.1.13.2.2 High-speed EOP

In high-speed signaling, a bit stuff error is intentionally generated to indicate EOP. A receiver is required to interpret any bit stuff error as an EOP.

For high-speed packets other than SOF's, the transmitted EOP delimiter is required to be an NRZ byte of 01111111 without bit stuffing. For example, if the last symbol prior to the EOP field is a J, this would lead to an EOP of KKKKKKKK.

For high-speed SOF's, the transmitted EOP delimiter is required to be 5 NRZ bytes without bit stuffing, consisting of 01111111 11111111 11111111 11111111 11111111. Thus if the last bit prior to the EOP field is a J, this would lead to 40 K's on the wire, at the end of which the lines must return to the high-speed Idle state. This extra EOP length is of no significance to a receiver; it is used for disconnect detection as discussed in Section 7.1.7.3.

A hub may add at most 4 random bits to the end of the EOP field when repeating a packet. Thus after 5 repeaters, a packet can have up to 20 random bits following the EOP field. A hub, however, must not corrupt any of the 8 (or 40 in the case of a SOF) required bits of the EOP field.

7.1.14 Hub Signaling Timings

This section describes low-speed, full-speed, and high-speed hub signaling timings.

7.1.14.1 Low-/full-speed Hub Signaling Timings

The propagation of a full-speed, differential data signal through a hub is shown in Figure 7-38. The downstream signaling is measured without a cable connected to the port and with the load used for measuring rise and fall times. The total delay through the upstream cable and hub electronics must be a maximum of 70 ns (THDD1). If the hub has a detachable USB cable, then the delay (THDD2) through hub electronics and the associated transmission line must be a maximum of 44 ns to allow for a maximum cable delay of 26 ns (TFSCBL). The delay through this hub is measured in both the upstream and downstream directions, as shown in Figure 7-38B, from data line crossover at the input port to data line crossover at the output port.

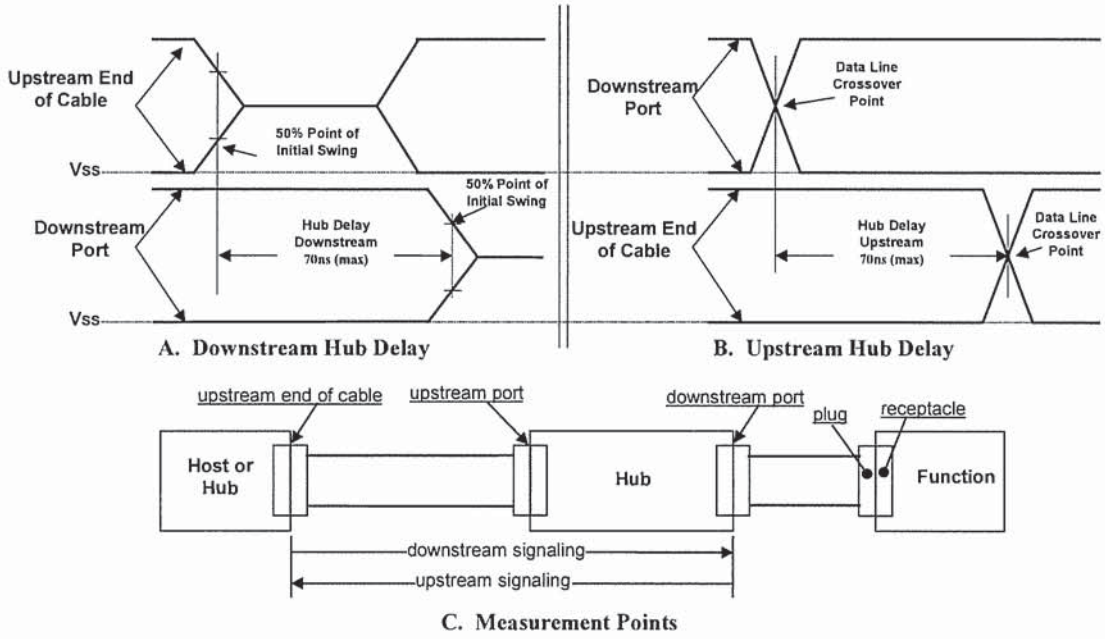


Figure 7-38. Hub Propagation Delay of Full-speed Differential Signals

Low-speed propagation delay for differential signals is measured in the same fashion as for full-speed signaling. The maximum low-speed hub delay is 300 ns (TLHDD). This allows for the slower low-speed buffer propagation delay and rise and fall times. It also provides time for the hub to re-clock the low-speed data in the upstream direction.

When the hub acts as a repeater, it must reproduce the received, full-speed signal accurately on its outputs. This means that for differential signals, the propagation delays of a J-to-K state transition must match closely to the delays of a K-to-J state transition. For full-speed propagation, the maximum difference allowed between these two delays (THDJ1) (see Figure 7-38 and Figure 7-52) for a hub plus cable is ±3.0 ns. Similarly, the difference in delay between any two J-to-K or K-to-J transitions through a hub (THDJ2) must be less than ±1.0 ns. For low-speed propagation in the downstream direction, the corresponding allowable jitter (TLDHJ1) is ±45 ns and (TLDHJ2) ±15 ns, respectively. For low-speed propagation in the upstream direction, the allowable jitter is ±45 ns in both cases (TLUHJ1 and TLUHJ2).

An exception to this case is the skew that can be introduced in the Idle-to-K state transition at SOP (TFSOP and TLSOP) (refer to Section 7.1.7.4). In this case, the delay to the opposite port includes the time to enable the output buffer. However, the delays should be closely matched to the normal hub delay and the maximum

additional delay difference over a normal J-to-K transition is ±5.0 ns. This limits the maximum distortion of the first bit in the packet.

Note: Because of this distortion of the SOP transition relative to the next K-to-J state transition, the first SYNC field bit should not be used to synchronize the receiver to the data stream.

The EOP must be propagated through a hub in the same way as the differential signaling. The propagation delay for sensing an SE0 must be no less than the greater of the J-to-K or K-to-J differential data delay (to avoid truncating the last data bit in a packet), but not more than 15 ns greater than the larger of these differential delays at full-speed and 200 ns at low-speed (to prevent creating a bit stuff error at the end of the packet). EOP delays are shown in Figure 7-53.

Because the sense levels for the SE0 state are not at the midpoint of the signal swing, the width of SE0 state will be changed as it passes through each hub. A hub may not change the width of the SE0 state in a full-speed EOP by more than ±15 ns (TFHESK), as measured by the difference of the leading edge and trailing edge delays of the SE0 state (see Figure 7-53). An SE0 from a low-speed device has long rise and fall times and is subject to greater skew, but these conditions exist only on the cable from the low-speed device to the port to which it is connected. Thereafter, the signaling uses full-speed buffers and their faster rise and fall times. The SE0 from the low-speed device cannot be changed by more than ±300 ns (TLHESK) as it passes through the hub to which the device is connected. This time allows for some signal conditioning in the low-speed transceiver to reduce its sensitivity to noise.

7.1.14.2 High-speed Hub Signaling Timings

When a hub acts as a repeater for high-speed data, the delay of the hub (THSHDD) must not exceed 36 high-speed bit times plus 4 ns (the trace delays allowed for the hub circuit board). This delay is measured from the last bit of the SYNC field at the input connector to the last bit of the SYNC field at the output connector.

A high-speed hub repeater must digitally resynchronize the buffered data, so there is no allowance for cumulative jitter (within a single packet) as a high-speed packet passes through multiple repeater stages. Within a single packet, the jitter must not exceed the eye pattern templates defined in Section 7.1.2 over a sliding window of 480 high-speed bit times.

Due to the data synchronization process, the propagation delay of a hub repeater is allowed to vary at most 5 high-speed bit times (THSHDV). The delay including this allowed variation must not exceed 36 high-speed bit times plus 4 ns. (This allows for some uncertainty as to when an incoming packet arrives at the hub with respect to the phase of the synchronization clock.)

7.1.15 Receiver Data Jitter

This section describes low-speed, full-speed, and high-speed receiver data jitter.

7.1.15.1 Low-/full-speed Receiver Data Jitter

The data receivers for all types of devices must be able to properly decode the differential data in the presence of jitter. The more of the bit cell that any data edge can occupy and still be decoded, the more reliable the data transfer will be. Data receivers are required to decode differential data transitions that occur in a window plus and minus a nominal quarter bit cell from the nominal (centered) data edge position. (A simple 4X over-sampling state machine DPLL can be built that satisfies these requirements.) This requirement is derived in Table 7-4 and Table 7-5. The tables assume a worst-case topology of five hubs between the host and device and the worst-case number of seven bits between transitions. The derived numbers are rounded up for ease of specification.

Jitter will be caused by the delay mismatches discussed above and by mismatches in the source and destination data rates (frequencies). The receive data jitter budgets for full- and low-speed are given in Table 7-4 and Table 7-5. These tables give the value and totals for each source of jitter for both consecutive (next) and paired transitions. Note that the jitter component related to the source or destination frequency tolerance has been allocated to the appropriate device (i.e., the source jitter includes bit shifts due to source frequency inaccuracy over the worst-case data transition interval). The output driver jitter can be traded off against the device clock accuracy in a particular implementation as long as the jitter specification is met.

The low-speed jitter budget table has an additional line in it because the jitter introduced by the hub to which the low-speed device is attached is different from all the other devices in the data path. The remaining devices operate with full-speed signaling conventions (though at low-speed data rate).

Table 7-4. Full-speed Jitter Budget

Jitter Source	Full-speed			
	Next Transition		Paired Transition	
	Each (ns)	Total (ns)	Each (ns)	Total (ns)
Source Driver Jitter	2.0	2.0	1.0	1.0
Source Frequency Tolerance (worst-case)	0.21/bit	1.5	0.21/bit	3.0
Source Jitter Total		**3.5**		**4.0**
Hub Jitter	3.0	15.0	1.0	5.0
Jitter Specification		**18.5**		**9.0**
Destination Frequency Tolerance	0.21/bit	1.5	0.21/bit	3.0
Receiver Jitter Budget		**20.0**		**12.0**

Table 7-5. Low-speed Jitter Budget

Jitter Source	Low-speed Upstream			
	Next Transition		Paired Transition	
	Each (ns)	Total (ns)	Each (ns)	Total (ns)
Function Driver Jitter	25.0	25.0	10.0	10.0
Function Frequency Tolerance (worst-case)	10.0/bit	70.0	10.0/bit	140.0
Source (Function) Jitter Total		**95.0**		**150.0**
Hub with Low-speed Device Jitter	45.0	45.0	45.0	45.0
Remaining (full-speed) Hubs' Jitter	3.0	12.0	1.0	4.0
Jitter Specification		**152.0**		**199.0**
Host Frequency Tolerance	1.7/bit	12.0	1.7/bit	24.0
Host Receiver Jitter Budget		**164.0**		**223.0**
	Low-speed Downstream			
	Next Transition		**Paired Transition**	
	Each (ns)	**Total (ns)**	**Each (ns)**	**Total (ns)**
Host Driver Jitter	2.0	2.0	1.0	1.0
Host Frequency Tolerance (worst-case)	1.7/bit	12.0	1.7/bit	24.0
Source (Host) Jitter Total		**14.0**		**25.0**
Hub with Low-speed Device Jitter	45.0	45.0	15.0	15.0
Remaining (full-speed) Hubs' Jitter	3.0	12.0	1.0	4.0
Jitter Spec		**71.0**		**44.0**
Function Frequency Tolerance	10.0/bit	70.0	10.0/bit	140.0
Function Receiver Jitter Budget		**141.0**		**184.0**

Note: This table describes the host transmitting at low-speed data rate using full-speed signaling to a low-speed device through the maximum number of hubs. When the host is directly connected to the low-speed device, it uses low-speed data rate and low-speed signaling, and the host has to meet the source jitter listed in the "Jitter Specification" row.

7.1.15.2 High-speed Receiver Data Jitter

A high-speed capable receiver must reliably recover high-speed data when the waveforms at its inputs conform to the receiver sensitivity eye pattern templates. The templates, which are called out in Section 7.1.2.2, specify the horizontal and vertical eye pattern opening over a 480 bit time sliding window over the duration of a packet. Thus, for example, a high-speed receiver within a function must reliably recover data with a peak to peak jitter of 30%, measured at its B receptacle (as described by Template 4).

Such conformance is tested using Test Mode Test_Packet, as defined in Section 7.1.20.

It is a recommended design guideline that a receiver's BER should be $<= 10^{-12}$ when the receiver sensitivity requirement is met.

7.1.16 Cable Delay

The maximum total one-way signal propagation delay allowed is 30 ns. The allocation for cable delay is 26 ns. A maximum delay of 3 ns is allowed from a Host or Hub Controller downstream facing transceiver to its exterior downstream facing connector, while a maximum delay of 1 ns is allowed from the upstream facing connector to the upstream facing transceiver of any device. For a standard USB detachable cable, the cable

delay is measured from the Series A connector pins to the Series B connector pins and is no more than 26 ns. For other cables, the delay is measured from the series A connector to the point where the cable is connected to the device. The cable delay must also be less than 5.2 ns per meter.

The maximum one-way data delay on a full-speed cable is measured as shown in Figure 7-39.

One-way cable delay for low-speed cables must be less than 18 ns. It is measured as shown in Figure 7-40.

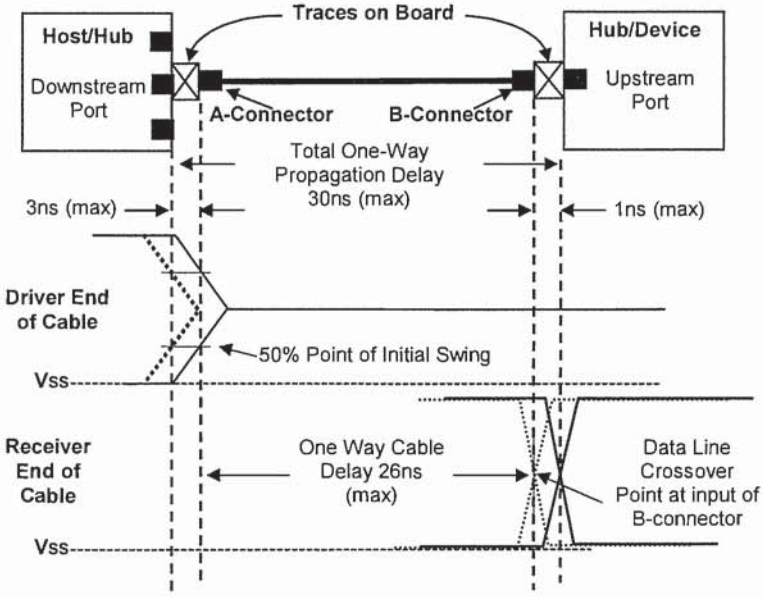


Figure 7-39. Full-speed Cable Delay

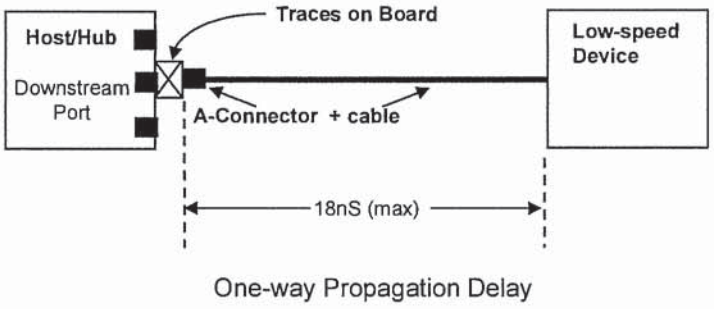


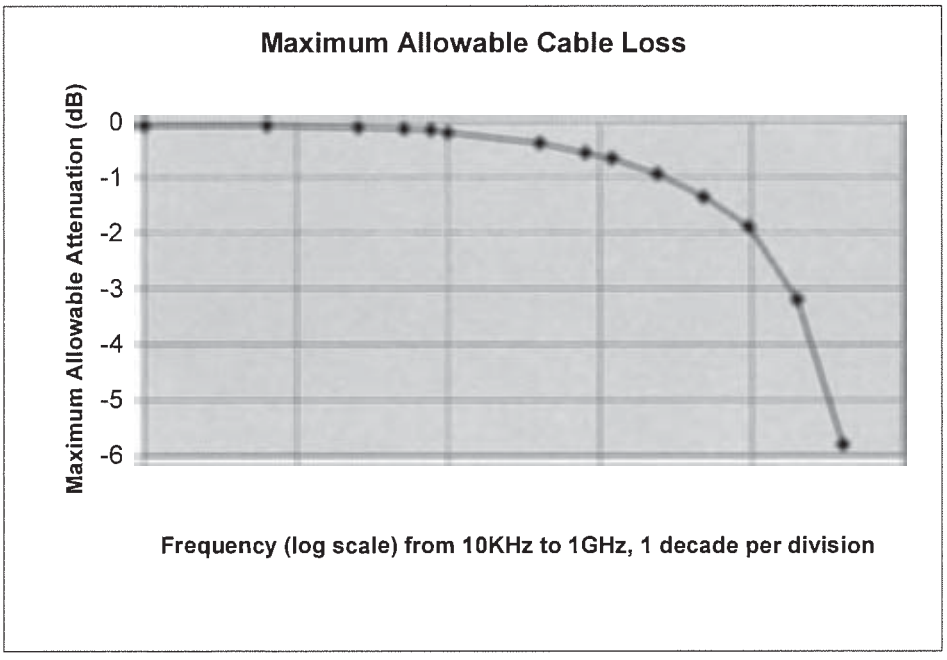
Figure 7-40. Low-speed Cable Delay

7.1.17 Cable Attenuation

USB cables must not exceed the loss figures shown in Table 7-6. Between the frequencies called out in the table, the cable loss should be no more than is shown in the accompanying graph.

Table 7-6. Maximum Allowable Cable Loss

Frequency (MHz)	Attenuation (maximum) dB/cable
0.064	0.08
0.256	0.11
0.512	0.13
0.772	0.15
1.000	0.20
4.000	0.39
8.000	0.57
12.000	0.67
24.000	0.95
48.000	1.35
96.000	1.9
200.00	3.2
400.00	5.8



7.1.18 Bus Turn-around Time and Inter-packet Delay

This section describes low-speed, full-speed, and high-speed bus turn-around time and inter-packet delay.

7.1.18.1 Low-/Full-Speed Bus Turn-around Time and Inter-packet Delay

Inter-packet delays are measured from the SE0-to-J transition at the end of the EOP to the J-to-K transition that starts the next packet.

A device must provide at least two bit times of inter-packet delay. The delay is measured at the responding device with a bit time defined in terms of the response. This provides adequate time for the device sending the EOP to drive J for one bit time and then turn off its output buffers.

The host must provide at least two bit times of J after the SE0 of an EOP and the start of a new packet (TIPD). If a function is expected to provide a response to a host transmission, the maximum inter-packet delay for a function or hub with a detachable (TRSPIPD1) cable is 6.5 bit times measured at the Series B receptacle. If the device has a captive cable, the inter-packet delay (TRSPIPD2) must be less than 7.5 bit times as measured at the Series A plug. These timings apply to both full-speed and low-speed devices and the bit times are referenced to the data rate of the packet.

The maximum inter-packet delay for a host response is 7.5 bit times measured at the host's port pins. There is no maximum inter-packet delay between packets in unrelated transactions.

7.1.18.2 High-Speed Bus Turn-around Time and Inter-packet Delay

High-speed inter-packet delays are measured from time when the line returns to a high-speed Idle State at the end of one packet to when the line leaves the high-speed Idle State at the start of the next packet.

When transmitting after receiving a packet, hosts and devices must provide an inter-packet delay of at least 8 bit times (THSIPDOD) measured at their A or B connectors (receptacles or plugs).

Additionally, if a host is transmitting two packets in a row, the inter-packet delay must be a minimum of 88 bit times (THSIPDSD), measured at the host's A receptacle. This will guarantee an inter-packet delay of at least 32 bit times at all devices (when receiving back to back packets). The maximum inter-packet delay provided by a host is 192 bit times within a transaction (THSRSPIPD1) measured at the A receptacle. When a host responds to a packet from a device, it will provide an inter-packet delay of at most 192 bit times measured at the A receptacle. There is no maximum inter-packet delay between packets in unrelated transactions.

When a device with a detachable cable responds to a packet from a host, it will provide an inter-packet delay of at most 192 bit times measured at the B receptacle. If the device has a captive cable, it will provide an inter-packet delay of at most 192 bit times plus 52 ns (2 times the max cable length) measured at the cable's A plug (THSRSPIPD2).

7.1.19 Maximum End-to-end Signal Delay

This section describes low-speed, full-speed, and high-speed end-to-end delay.

7.1.19.1 Low-/full-speed End-to-end Signal Delay

A device expecting a response to a transmission will invalidate the transaction if it does not see the start-of-packet (SOP) transition within the timeout period after the end of the transmission (after the SE0-to-J state transition in the EOP). This can occur between an IN token and the following data packet or between a data packet and the handshake packet (refer to Chapter 8). The device expecting the response will not time out before 16 bit times but will timeout before 18 bit times (measured at the data pins of the device from the SE0-to-J transition at the end of the EOP). The host will wait at least 18 bit times for a response to start before it will start a new transaction.

Figure 7-41 depicts the configuration of six signal hops (cables) that results in allowable worst-case signal delay. The maximum propagation delay from the upstream end of a hub's cable to any downstream facing connector on that hub is 70 ns.

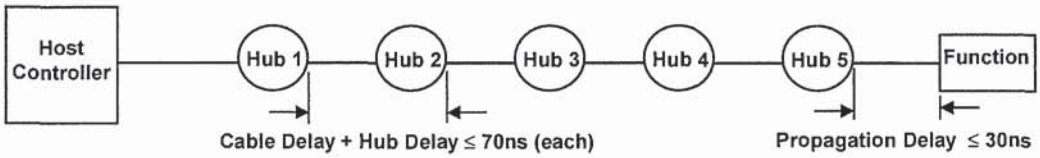


Figure 7-41. Worst-case End-to-end Signal Delay Model for Low-/full-speed

7.1.19.2 High-Speed End-to-end Delay

A high-speed host or device expecting a response to a transmission must not timeout the transaction if the inter-packet delay is less than 736 bit times, and it must timeout the transaction if no signaling is seen within 816 bit times.

These timeout limits allow a response to be seen even for the worst-case round trip signal delay. In high-speed mode, the worst-case round trip signal delay model is the sum of the following components:

12 max length cable delays (6 cables)	= 312 ns
10 max delay hubs (5 hubs)	= 40 ns + 360 bit times
1 max device response time	= 192 bit times
Worst-case round trip delay	= 352 ns +552 bit times = 721 bit times

7.1.20 Test Mode Support

To facilitate compliance testing, host controllers, hubs, and high-speed capable functions must support the following test modes:

- Test mode Test_SE0_NAK: Upon command, a port's transceiver must enter the high-speed receive mode and remain in that mode until the exit action is taken. This enables the testing of output impedance, low level output voltage, and loading characteristics. In addition, while in this mode, upstream facing ports (and only upstream facing ports) must respond to any IN token packet with a NAK handshake (only if the packet CRC is determined to be correct) within the normal allowed device response time. This enables testing of the device squelch level circuitry and, additionally, provides a general purpose stimulus/response test for basic functional testing.
- Test mode Test_J: Upon command, a port's transceiver must enter the high-speed J state and remain in that state until the exit action is taken. This enables the testing of the high output drive level on the D+ line.
- Test mode Test_K: Upon command, a port's transceiver must enter the high-speed K state and remain in that state until the exit action is taken. This enables the testing of the high output drive level on the D- line.
- Test mode Test_Packet: Upon command, a port must repetitively transmit the following test packet until the exit action is taken. This enables the testing of rise and fall times, eye patterns, jitter, and any other dynamic waveform specifications.

  The test packet is made up by concatenating the following strings. (Note: For J/K NRZI data, and for NRZ data, the bit on the left is the first one transmitted. "S" indicates that a bit stuff occurs, which inserts an "extra" NRZI data bit. "* N" is used to indicate N occurrences of a string of bits or symbols.)

NRZI Symbols (Fields)	NRZ Bit Strings	Number of NRZ Bits
{KJ * 15}, KK (SYNC)	{00000000 * 3}, 00000001	32
KKJKJKKK (DATA0 PID)	11000011	8
JKJKJKJK * 9	00000000 * 9	72
JJKKJJKK * 8	01010101 * 8	64
JJJJKKKK * 8	01110111 * 8	64
JJJJJJJKKKKKKK * 8	0, {111111S *15}, 111111	97
JJJJJJJK * 8	S, 111111S, {0111111S * 7}	55
{JKKKKKKK * 10}, JK	00111111, {S0111111 * 9}, S0	72
JJJKKKJJKKKKJKKK (CRC16)	0110110101110011	16
JJJJJJJJ (EOP)	01111111	8

A port in Test_Packet mode must send this packet repetitively. The inter-packet timing must be no less than the minimum allowable inter-packet gap as defined in Section 7.1.18 and no greater than 125 µs.

- Test mode Test_Force_Enable: Upon command, downstream facing hub ports (and only downstream facing hub ports) must be enabled in high-speed mode, even if there is no device attached. Packets arriving at the hub's upstream facing port must be repeated on the port which is in this test mode. This enables testing of the hub's disconnect detection; the disconnect detect bit can be polled while varying the loading on the port, allowing the disconnect detection threshold voltage to be measured.

Test Mode Entry and Exit

Test mode of a port is entered by using a device specific standard request (for an upstream facing port) or a port specific hub class request (for a downstream facing port). The device standard request SetFeature(TEST_MODE) is defined in Section 9.4.9. The hub class request SetPortFeature(PORT_TEST) is defined in Section 11.24.2.13. All high-speed capable devices/hubs must support these requests. These requests are not supported for non-high-speed devices.

The transition to test mode must be complete no later than 3 ms after the completion of the status stage of the request.

For an upstream facing port, the exit action is to power cycle the device. For a downstream facing port, the exit action is to reset the hub, as defined in Section 11.24.2.13.

7.2 Power Distribution

This section describes the USB power distribution specification.

7.2.1 Classes of Devices

The power source and sink requirements of different device classes can be simplified with the introduction of the concept of a unit load. A unit load is defined to be 100 mA. The number of unit loads a device can draw is an absolute maximum, not an average over time. A device may be either low-power at one unit load or high-power, consuming up to five unit loads. All devices default to low-power. The transition to high-power is under software control. It is the responsibility of software to ensure adequate power is available before allowing devices to consume high-power.

The USB supports a range of power sourcing and power consuming agents; these include the following:

- **Root port hubs**: Are directly attached to the USB Host Controller. Hub power is derived from the same source as the Host Controller. Systems that obtain operating power externally, either AC or DC, must supply at least five unit loads to each port. Such ports are called high-power ports. Battery-powered systems may supply either one or five unit loads. Ports that can supply only one unit load are termed low-power ports.
- **Bus-powered hubs**: Draw all of their power for any internal functions and downstream facing ports from VBUS on the hub's upstream facing port. Bus-powered hubs may only draw up to one unit load upon power-up and five unit loads after configuration. The configuration power is split between allocations to the hub, any non-removable functions and the external ports. External ports in a bus-powered hub can supply only one unit load per port regardless of the current draw on the other ports of that hub. The hub must be able to supply this port current when the hub is in the Active or Suspend state.
- **Self-powered hubs**: Power for the internal functions and downstream facing ports does not come from VBUS. However, the USB interface of the hub may draw up to one unit load from VBUS on its upstream facing port to allow the interface to function when the remainder of the hub is powered down. Hubs that obtain operating power externally (from the USB) must supply five unit loads to each port. Battery-powered hubs may supply either one or five unit loads per port.
- **Low-power bus-powered functions**: All power to these devices comes from VBUS. They may draw no more than one unit load at any time.
- **High-power bus-powered functions**: All power to these devices comes from VBUS. They must draw no more than one unit load upon power-up and may draw up to five unit loads after being configured.
- **Self-powered functions**: May draw up to one unit load from VBUS to allow the USB interface to function when the remainder of the function is powered down. All other power comes from an external (to the USB) source.

No device shall supply (source) current on VBUS at its upstream facing port at any time. From VBUS on its upstream facing port, a device may only draw (sink) current. They may not provide power to the pull-up resistor on D+/D- unless VBUS is present (see Section 7.1.5). When VBUS is removed, the device must remove power from the D+/D- pull-up resistor within 10 seconds. On power-up, a device needs to ensure that its upstream facing port is not driving the bus, so that the device is able to receive the reset signaling. Devices must also ensure that the maximum operating current drawn by a device is one unit load, until configured. Any device that draws power from the bus must be able to detect lack of activity on the bus, enter the Suspend state, and reduce its current consumption from VBUS (refer to Section 7.2.3 and Section 9.2.5.1).

7.2.1.1 Bus-powered Hubs

Bus-powered hub power requirements can be met with a power control circuit such as the one shown in Figure 7-42. Bus-powered hubs often contain at least one non-removable function. Power is always available to the hub's controller, which permits host access to power management and other configuration registers during the enumeration process. A non-removable function(s) may require that its power be switched, so that upon power-up, the entire device (hub and non-removable functions) draws no more than one unit load. Power switching on any non-removable function may be implemented either by removing its power or by shutting off the clock. Switching on the non-removable function is not required if the aggregate power drawn by it and the Hub Controller is less than one unit load. However, as long as the hub port associated with the function is in the Power-off state, the function must be logically reset and the device must appear to be not connected. The total current drawn by a bus-powered device is the sum of the current to the Hub Controller, any non-removable function(s), and the downstream facing ports.

Figure 7-42 shows the partitioning of power based upon the maximum current draw (from upstream) of five unit loads: one unit load for the Hub Controller and the non-removable function and one unit load for each of the external downstream facing ports. If more than four external ports are required, then the hub will need to be self-powered. If the non-removable function(s) and Hub Controller draw more than one unit load, then the number of external ports must be appropriately reduced. Power control to a bus-powered hub may require a regulator. If present, the regulator is always enabled to supply the Hub Controller. The regulator can also power the non-removable functions(s). Inrush current limiting must also be incorporated into the regulator subsystem.

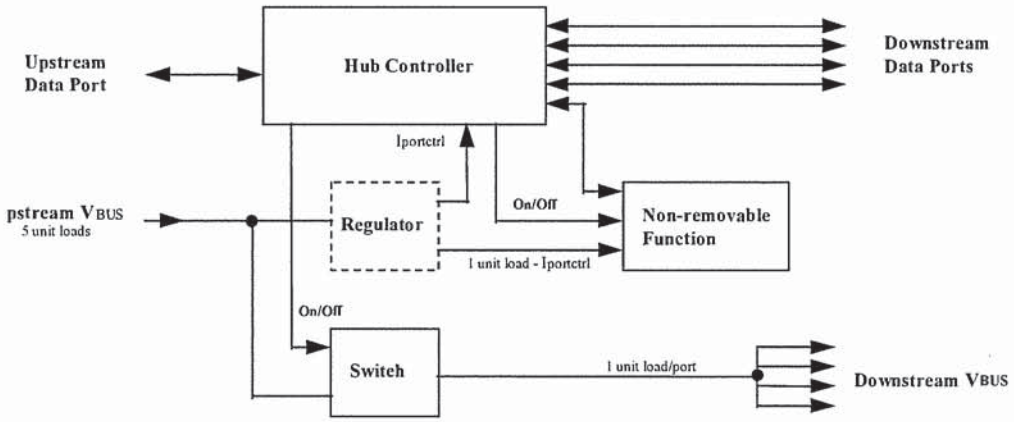


Figure 7-42. Compound Bus-powered Hub

Power to external downstream facing ports of a bus-powered hub must be switched. The Hub Controller supplies a software controlled on/off signal from the host, which is in the "off" state when the device is powered up or after reset signaling. When switched to the "on" state, the switch implements a soft turn-on function that prevents excessive transient current from being drawn from upstream. The voltage drop across the upstream cable, connectors, and switch in a bus-powered hub must not exceed 350 mV at maximum rated current.

7.2.1.2 Self-powered Hubs

Self-powered hubs have a local power supply that furnishes power to any non-removable functions and to all downstream facing ports, as shown in Figure 7-43. Power for the Hub Controller, however, may be supplied from the upstream VBUS (a "hybrid" powered hub) or the local power supply. The advantage of supplying the Hub Controller from the upstream supply is that communication from the host is possible even if the device's power supply remains off. This makes it possible to differentiate between a disconnected and an unpowered device. If the hub draws power for its upstream facing port from VBUS, it may not draw more than one unit load.

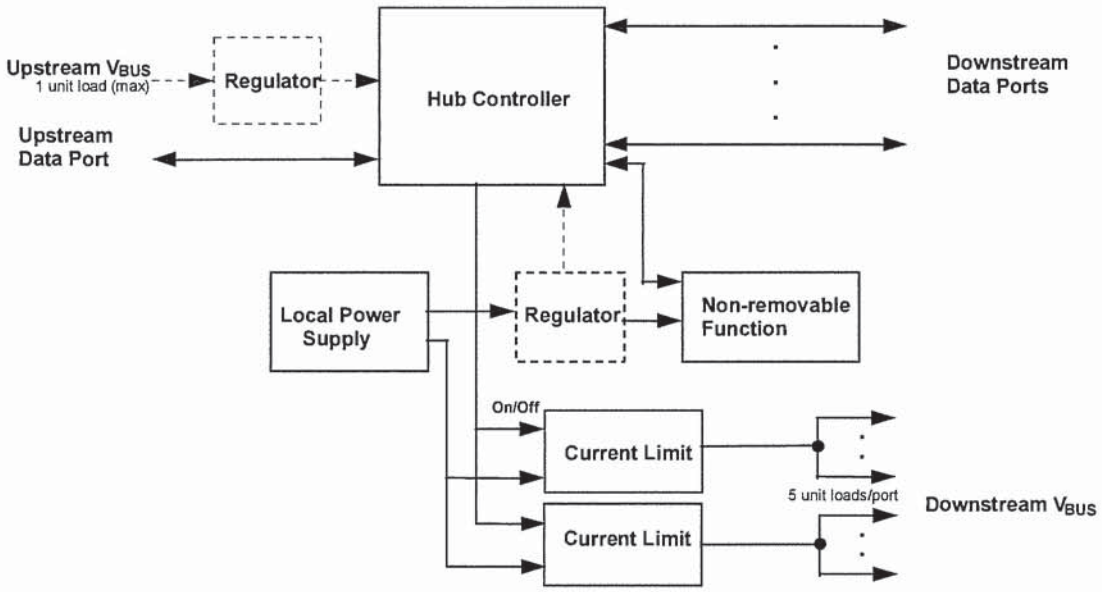


Figure 7-43. Compound Self-powered Hub

The number of ports that can be supported is limited only by the address capability of the hub and the local supply.

Self-powered hubs may experience loss of power. This may be the result of disconnecting the power cord or exhausting the battery. Under these conditions, the hub may force a re-enumeration of itself as a bus-powered hub. This requires the hub to implement port power switching on all external ports. When power is lost, the hub must ensure that upstream current does not exceed low-power. All the rules of a bus-powered hub then apply.

7.2.1.2.1 Over-current Protection

The host and all self-powered hubs must implement over-current protection for safety reasons, and the hub must have a way to detect the over-current condition and report it to the USB software. Should the aggregate current drawn by a gang of downstream facing ports exceed a preset value, the over-current protection circuit removes or reduces power from all affected downstream facing ports. The over-current condition is reported through the hub to Host Controller, as described in Section 11.12.5. The preset value cannot exceed 5.0 A and must be sufficiently above the maximum allowable port current such that transient currents (e.g., during power up or dynamic attach or reconfiguration) do not trip the over-current protector. If an over-current condition occurs on any port, subsequent operation of the USB is not guaranteed, and once the condition is removed, it may be necessary to reinitialize the bus as would be done upon power-up. The over-current limiting mechanism must be resettable without user mechanical intervention. Polymeric PTCs and solid-state switches are examples of methods, which can be used for over-current limiting.

7.2.1.3 Low-power Bus-powered Functions

A low-power function is one that draws up to one unit load from the USB cable when operational. Figure 7-44 shows a typical bus-powered, low-power function, such as a mouse. Low-power regulation can be integrated into the function silicon. Low-power functions must be capable of operating with input VBUS voltages as low as 4.40 V, measured at the plug end of the cable.

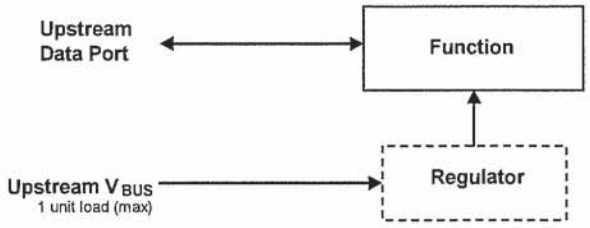


Figure 7-44. Low-power Bus-powered Function

7.2.1.4 High-power Bus-powered Functions

A function is defined as being high-power if, when fully powered, it draws over one but no more than five unit loads from the USB cable. A high-power function requires staged switching of power. It must first come up in a reduced power state of less than one unit load. At bus enumeration time, its total power requirements are obtained and compared against the available power budget. If sufficient power exists, the remainder of the function may be powered on. A typical high-power function is shown in Figure 7-45. The function's electronics have been partitioned into two sections. The function controller contains the minimum amount of circuitry necessary to permit enumeration and power budgeting. The remainder of the function resides in the function block. High-power functions must be capable of operating in their low-power (one unit load) mode with an input voltage as low as 4.40 V, so that it may be detected and enumerated even when plugged into a bus-powered hub. They must also be capable of operating at full power (up to five unit loads) with a VBUS voltage of 4.75 V, measured at the upstream plug end of the cable.

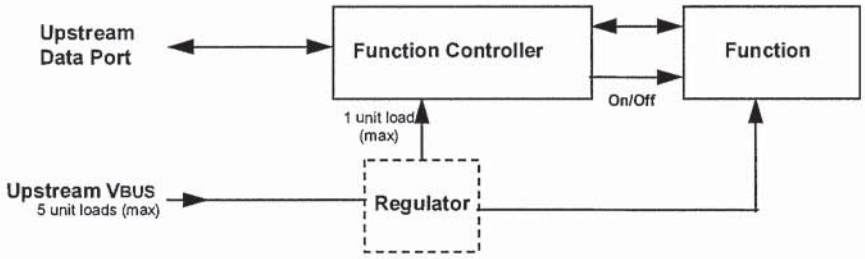


Figure 7-45. High-power Bus-powered Function

7.2.1.5 Self-powered Functions

Figure 7-46 shows a typical self-powered function. The function controller is powered either from the upstream bus via a low-power regulator or from the local power supply. The advantage of the former scheme is that it permits detection and enumeration of a self-powered function whose local power supply is turned off. The maximum upstream power that the function controller can draw is one unit load, and the regulator block must implement inrush current limiting. The amount of power that the function block may draw is limited only by the local power supply. Because the local power supply is not required to power any downstream bus ports, it does not need to implement current limiting, soft start, or power switching.

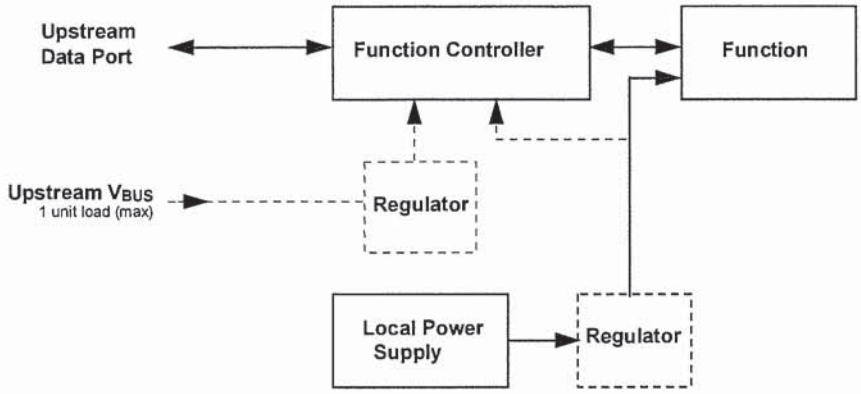


Figure 7-46. Self-powered Function

7.2.2 Voltage Drop Budget

The voltage drop budget is determined from the following:

- The voltage supplied by high-powered hub ports is 4.75 V to 5.25 V.
- The voltage supplied by low-powered hub ports is 4.4 V to 5.25 V.
- Bus-powered hubs can have a maximum drop of 350 mV from their cable plug (where they attach to a source of power) to their output port connectors (where they supply power).
- The maximum voltage drop (for detachable cables) between the A-series plug and B-series plug on VBUS is 125 mV (VBUSD).
- The maximum voltage drop for all cables between upstream and downstream on GND is 125 mV (VGNDD).
- All hubs and functions must be able to provide configuration information with as little as 4.40 V at the connector end of their upstream cables. Only low-power functions need to be operational with this minimum voltage.
- Functions drawing more than one unit load must operate with a 4.75 V minimum input voltage at the connector end of their upstream cables.

Figure 7-47 shows the minimum allowable voltages in a worst-case topology consisting of a bus-powered hub driving a bus-powered function.

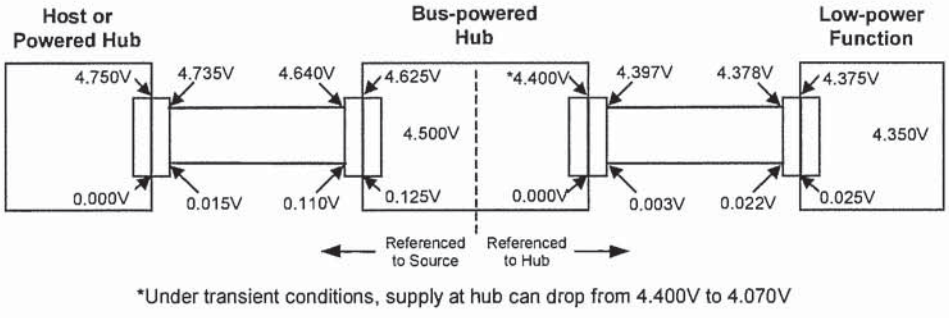


Figure 7-47. Worst-case Voltage Drop Topology (Steady State)

7.2.3 Power Control During Suspend/Resume

Suspend current is a function of unit load allocation. All USB devices initially default to low-power. Low-power devices or high-power devices operating at low-power are limited to 500 µA of suspend current. If the device is configured for high-power and enabled as a remote wakeup source, it may draw up to 2.5 mA during suspend. When computing suspend current, the current from VBUS through the bus pull-up and pull-down resistors must be included. Configured bus-powered hubs may also consume a maximum of 2.5 mA, with 500 µA allocated to each available external port and the remainder available to the hub and its internal functions. If a hub is not configured, it is operating as a low-power device and must limit its suspend current to 500 µA.

While in the Suspend state, a device may briefly draw more than the average current. The amplitude of the current spike cannot exceed the device power allocation 100 mA (or 500 mA). A maximum of 1.0 second is allowed for an averaging interval. The average current cannot exceed the average suspend current limit (ICCSH or ICCSL, see Table 7-7) during any 1.0-second interval (TSUSAVG1). The profile of the current spike is restricted so the transient response of the power supply (which may be an efficient, low-capacity, trickle power supply) is not overwhelmed. The rising edge of the current spike must be no more than 100 mA/µs. Downstream facing ports must be able to absorb the 500 mA peak current spike and meet the voltage droop requirements defined for inrush current during dynamic attach (see Section 7.2.4.1). Figure 7-48 illustrates a typical example profile for an averaging interval. If the supply to the pull-up resistor on D+/D- is derived from VBUS, then the suspend current will never go to zero because the pull-up and pull-down resistors will always draw power.

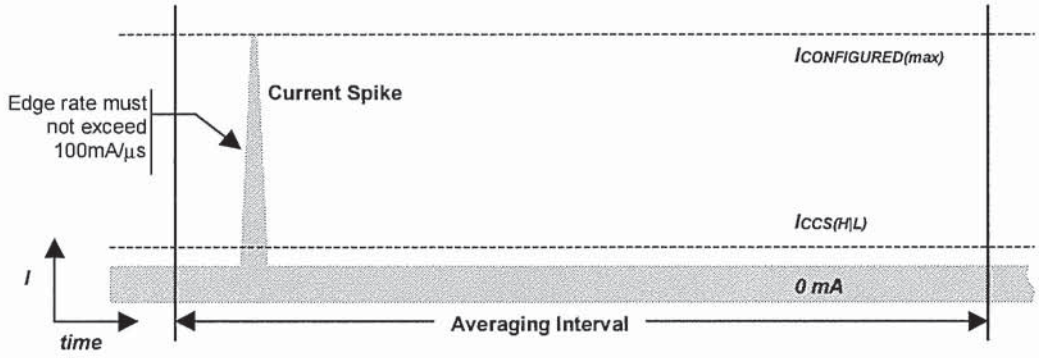


Figure 7-48. Typical Suspend Current Averaging Profile

Devices are responsible for handling the bus voltage reduction due to the inductive and resistive effects of the cable. When a hub is in the Suspend state, it must still be able to provide the maximum current per port (one unit load of current per port for bus-powered hubs and five unit loads per port for self-powered hubs). This is necessary to support remote wakeup-capable devices that will power-up while the remainder of the system is still suspended. Such devices, when enabled to do remote wakeup, must drive resume signaling upstream within 10 ms of starting to draw the higher, non-suspend current. Devices not capable of remote wakeup must draw the higher current only when not suspended.

When devices wakeup, either by themselves (remote wakeup) or by seeing resume signaling, they must limit the inrush current on VBUS. The target maximum droop in the hub VBUS is 330 mV. The device must have sufficient on-board bypass capacitance or a controlled power-on sequence such that the current drawn from the hub does not exceed the maximum current capability of the port at any time while the device is waking up.

7.2.4 Dynamic Attach and Detach

The act of plugging or unplugging a hub or function must not affect the functionality of another device on other segments of the network. Unplugging a function will stop the transaction between that function and the host. However, the hub to which this function was attached will recover from this condition and will alert the host that the port has been disconnected.

7.2.4.1 Inrush Current Limiting

When a function or hub is plugged into the network, it has a certain amount of on-board capacitance between VBUS and ground. In addition, the regulator on the device may supply current to its output bypass capacitance and to the function as soon as power is applied. Consequently, if no measures are taken to prevent it, there could be a surge of current into the device which might pull the VBUS on the hub below its minimum operating level. Inrush currents can also occur when a high-power function is switched into its high-power mode. This problem must be solved by limiting the inrush current and by providing sufficient capacitance in each hub to prevent the power supplied to the other ports from going out of tolerance. An additional motivation for limiting inrush current is to minimize contact arcing, thereby prolonging connector contact life.

The maximum droop in the hub VBUS is 330 mV, or about 10% of the nominal signal swing from the function. In order to meet this requirement, the following conditions must be met:

- The maximum load (CRPB) that can be placed at the downstream end of a cable is 10 µF in parallel with 44 Ω. The 10 µF capacitance represents any bypass capacitor directly connected across the VBUS lines in the function plus any capacitive effects visible through the regulator in the device. The 44 Ω resistance represents one unit load of current drawn by the device during connect.
- If more bypass capacitance is required in the device, then the device must incorporate some form of VBUS surge current limiting, such that it matches the characteristics of the above load.
- The hub downstream facing port VBUS power lines must be bypassed (CHPB) with no less than 120 µF of low-ESR capacitance per hub. Standard bypass methods should be used to minimize inductance and resistance between the bypass capacitors and the connectors to reduce droop. The bypass capacitors themselves should have a low dissipation factor to allow decoupling at higher frequencies.

The upstream facing port of a hub is also required to meet the above requirements. Furthermore, a bus-powered hub must provide additional surge limiting in the form of a soft-start circuit when it enables power to its downstream facing ports.

A high-power bus-powered device that is switching from a lower power configuration to a higher power configuration must not cause droop > 330 mV on the VBUS at its upstream hub. The device can meet this by ensuring that changes in the capacitive load it presents do not exceed 10 µF.

Signal pins are protected from excessive currents during dynamic attach by being recessed in the connector such that the power pins make contact first. This guarantees that the power rails to the downstream device are referenced before the signal pins make contact. In addition, the signal lines are in a high-impedance state during connect, so that no current flows for standard signal levels.

7.2.4.2 Dynamic Detach

When a device is detached from the network with power flowing in the cable, the inductance of the cable will cause a large flyback voltage to occur on the open end of the device cable. This flyback voltage is not destructive. Proper bypass measures on the hub ports will suppress any coupled noise. The frequency range of this noise is inversely dependent on the length of the cable, to a maximum of 60 MHz for a one-meter cable. This will require some low capacitance, very low inductance bypass capacitors on each hub port connector. The flyback voltage and the noise it creates is also moderated by the bypass capacitance on the device end of the cable. Also, there must be some minimum capacitance on the device end of the cable to ensure that the

inductive flyback on the open end of the cable does not cause the voltage on the device end to reverse polarity. A minimum of 1.0 µF is recommended for bypass across VBUS.

7.3 Physical Layer

The physical layer specifications are described in the following subsections.

7.3.1 Regulatory Requirements

All USB devices should be designed to meet the applicable regulatory requirements.

7.3.2 Bus Timing/Electrical Characteristics

Table 7-7. DC Electrical Characteristics

Parameter	Symbol	Conditions	Min.	Max.	Units
Supply Voltage:					
High-power Port	VBUS	Note 2, Section 7.2.1	4.75	5.25	V
Low-power Port	VBUS	Note 2, Section 7.2.1	4.40	5.25	V
Supply Current:					
High-power Hub Port (out)	ICCPRT	Section 7.2.1	500		mA
Low-power Hub Port (out)	ICCUPT	Section 7.2.1	100		mA
High-power Function (in)	ICCHPF	Section 7.2.1		500	mA
Low-power Function (in)	ICCLPF	Section 7.2.1		100	mA
Unconfigured Function/Hub (in)	ICCINIT	Section 7.2.1.4		100	mA
Suspended High-power Device	ICCSH	Section 7.2.3; Note 15		2.5	mA
Suspended Low-power Device	ICCSL	Section 7.2.3		500	µA
Input Levels for Low-/full-speed:					
High (driven)	VIH	Note 4, Section 7.1.4	2.0		V
High (floating)	VIHZ	Note 4, Section 7.1.4	2.7	3.6	V
Low	VIL	Note 4, Section 7.1.4		0.8	V
Differential Input Sensitivity	VDI	|(D+)-(D-)|; Figure 7-19; Note 4	0.2		V
Differential Common Mode Range	VCM	Includes VDI range; Figure 7-19; Note 4	0.8	2.5	V
Input Levels for High-speed:					
High-speed squelch detection threshold (differential signal amplitude)	VHSSQ	Section 7.1.7.2 (specification refers to differential signal amplitude)	100	150	mV

Table 7-7. DC Electrical Characteristics (Continued)

Parameter	Symbol	Conditions	Min.	Max.	Units
High speed disconnect detection threshold (differential signal amplitude)	V_{HSDSC}	Section 7.1.7.2 (specification refers to differential signal amplitude)	525	625	mV
High-speed differential input signaling levels		Section 7.1.7.2 Specified by eye pattern templates			
High-speed data signaling common mode voltage range (guideline for receiver)	V_{HSCM}	Section 7.1.4.2	-50	500	mV
Output Levels for Low-/full-speed:					
Low	V_{OL}	Note 4, 5, Section 7.1.1	0.0	0.3	V
High (Driven)	V_{OH}	Note 4, 6, Section 7.1.1	2.8	3.6	V
SE1	V_{OSE1}	Section 7.1.1	0.8		V
Output Signal Crossover Voltage	V_{CRS}	Measured as in Figure 7-8; Note 10	1.3	2.0	V
Output Levels for High-speed:					
High-speed idle level	V_{HSOI}	Section 7.1.7.2	-10.0	10.0	mV
High-speed data signaling high	V_{HSOH}	Section 7.1.7.2	360	440	mV
High-speed data signaling low	V_{HSOL}	Section 7.1.7.2	-10.0	10.0	mV
Chirp J level (differential voltage)	V_{CHIRPJ}	Section 7.1.7.2	700	1100	mV
Chirp K level (differential voltage)	V_{CHIRPK}	Section 7.1.7.2	-900	-500	mV
Decoupling Capacitance:					
Downstream Facing Port Bypass Capacitance (per hub)	C_{HPB}	VBUS to GND, Section 7.2.4.1	120		µF
Upstream Facing Port Bypass Capacitance	C_{RPB}	VBUS to GND; Note 9, Section 7.2.4.1	1.0	10.0	µF
Input Capacitance for Low-/full-speed:					
Downstream Facing Port	C_{IND}	Note 2; Section 7.1.6.1		150	pF
Upstream Facing Port (w/o cable)	C_{INUB}	Note 3; Section 7.1.6.1		100	pF
Transceiver edge rate control capacitance	C_{EDGE}	Section 7.1.6.1		75	pF

Table 7-7. DC Electrical Characteristics (Continued)

Parameter	Symbol	Conditions	Min.	Max.	Units
Input Impedance for High-speed:					
TDR spec for high-speed termination		Section 7.1.6.2			
Terminations:					
Bus Pull-up Resistor on Upstream Facing Port	RPU	1.5 kΩ ±5% Section 7.1.5	1.425	1.575	kΩ
Bus Pull-down Resistor on Downstream Facing Port	RPD	15 kΩ ±5% Section 7.1.5	14.25	15.75	kΩ
Input impedance exclusive of pullup/pulldown (for low-/full-speed)	ZINP	Section 7.1.6	300		kΩ
Termination voltage for upstream facing port pullup (RPU)	VTERM	Section 7.1.5	3.0	3.6	V
Terminations in High-speed:					
Termination voltage in high-speed	VHSTERM	Section 7.1.6.2	-10	10	mV

Table 7-8. High-speed Source Electrical Characteristics

Parameter	Symbol	Conditions	Min.	Max.	Units
Driver Characteristics:					
Rise Time (10% - 90%)	THSR	Section 7.1.2	500		ps
Fall Time (10% - 90%)	THSF	Section 7.1.2	500		ps
Driver waveform requirements		Specified by eye pattern templates in Section 7.1.2			
Driver Output Resistance (which also serves as high-speed termination)	ZHSDRV	Section 7.1.1.1	40.5	49.5	Ω
Clock Timings:					
High-speed Data Rate	THSDRAT	Section 7.1.11	479.760	480.240	Mb/s
Microframe Interval	THSFRAM	Section 7.1.12	124.9375	125.0625	µs
Consecutive Microframe Interval Difference	THSRFI	Section 7.1.12		4 high-speed bit times	
High-speed Data Timings:					
Data source jitter		Source and receiver jitter specified by the eye pattern templates in Section 7.1.2.2			
Receiver jitter tolerance					

Table 7-9. Full-speed Source Electrical Characteristics

Parameter	Symbol	Conditions	Min.	Max.	Units
Driver Characteristics:					
Rise Time	T_{FR}	Figure 7-8; Figure 7-9	4	20	ns
Fall Time	T_{FF}	Figure 7-8; Figure 7-9	4	20	ns
Differential Rise and Fall Time Matching	T_{FRFM}	(T_{FR}/T_{FF}) Note 10, Section 7.1.2	90	111.11	%
Driver Output Resistance for driver which is not high-speed capable	Z_{DRV}	Section 7.1.1.1	28	44	Ω
Clock Timings:					
Full-speed Data Rate for hubs and devices which are high-speed capable	$T_{FDRATHS}$	Average bit rate, Section 7.1.11	11.9940	12.0060	Mb/s
Full-speed Data Rate for devices which are not high-speed capable	T_{FDRATE}	Average bit rate, Section 7.1.11	11.9700	12.0300	Mb/s
Frame Interval	T_{FRAME}	Section 7.1.12	0.9995	1.0005	ms
Consecutive Frame Interval Jitter	T_{RFI}	No clock adjustment Section 7.1.12		42	ns
Full-speed Data Timings:					
Source Jitter Total (including frequency tolerance): To Next Transition For Paired Transitions	 T_{DJ1} T_{DJ2}	Note 7, 8, 12, 10; Measured as in Figure 7-49;	 -3.5 -4	 3.5 4	 ns ns
Source Jitter for Differential Transition to SE0 Transition	T_{FDEOP}	Note 8; Figure 7-50; Note 11	-2	5	ns
Receiver Jitter: To Next Transition For Paired Transitions	 T_{JR1} T_{JR2}	Note 8; Figure 7-51	 -18.5 -9	 18.5 9	 ns ns
Source SE0 interval of EOP	T_{FEOPT}	Figure 7-50	160	175	ns
Receiver SE0 interval of EOP	T_{FEOPR}	Note 13; Section 7.1.13.2; Figure 7-50	82		ns
Width of SE0 interval during differential transition	T_{FST}	Section 7.1.4		14	ns

Table 7-10. Low-speed Source Electrical Characteristics

Parameter	Symbol	Conditions	Min.	Max.	Units
Driver Characteristics:					
Transition Time:					
Rise Time	TLR	Measured as in Figure 7-8	75	300	ns
Fall Time	TLF		75	300	ns
Rise and Fall Time Matching	TLRFM	(TLR/TLF) Note 10	80	125	%
Upstream Facing Port (w/cable, low-speed only)	CLINUA	Note 1; Section 7.1.6	200	450	pF
Clock Timings:					
Low-speed Data Rate for hubs which are high-speed capable	TLDRATHS	Section 7.1.11	1.49925	1.50075	Mb/s
Low-speed Data Rate for devices which are not high-speed capable	TLDRATE	Section 7.1.11	1.4775	1.5225	Mb/s
Low-speed Data Timings:					
Upstream facing port source Jitter Total (including frequency tolerance):		Note 7, 8; Figure 7-49			
To Next Transition	TUDJ1		-95	95	ns
For Paired Transitions	TUDJ2		-150	150	ns
Upstream facing port source Jitter for Differential Transition to SE0 Transition	TLDEOP	Note 8; Figure 7-50; Note 11	-40	100	ns
Upstream facing port differential Receiver Jitter:		Note 8; Figure 7-51			
To Next Transition	TDJR1		-75	75	ns
For Paired Transitions	TDJR2		-45	45	ns
Downstream facing port source Jitter Total (including frequency tolerance):		Note 7, 8; Figure 7-49			
To Next Transition	TDDJ1		-25	25	ns
For Paired Transitions	TDDJ2		-14	14	ns
Downstream facing port source Jitter for Differential Transition to SE0 Transition		Note 8; Figure 7-50; Note 11			ns
Downstream facing port Differential Receiver Jitter:		Note 8; Figure 7-50			
To Next Transition	TUJR1		-152	152	ns
For Paired Transitions	TUJR2		-200	200	ns

Table 7-10. Low-speed Source Electrical Characteristics (Continued)

Parameter	Symbol	Conditions	Min.	Max.	Units
Source SE0 interval of EOP	TLEOPT	Figure 7-50	1.25	1.50	µs
Receiver SE0 interval of EOP	TLEOPR	Note 13; Section 7.1.13.2; Figure 7-50	670		ns
Width of SE0 interval during differential transition	TLST	Section 7.1.4		210	ns

Table 7-11. Hub/Repeater Electrical Characteristics

Parameter	Symbol	Conditions	Min.	Max.	Units
Full-speed Hub Characteristics (as measured at connectors):					
Driver Characteristics: (Refer to Table 7-9)		Upstream facing port and downstream facing ports configured as full-speed			
Hub Differential Data Delay:		Note 7, 8			
(with cable)	THDD1	Figure 7-52A		70	ns
(without cable)	THDD2	Figure 7-52B		44	ns
Hub Differential Driver Jitter: (including cable)		Note 7, 8; Figure 7-52, Section 7.1.14			
To Next Transition	THDJ1		-3	3	ns
For Paired Transitions	THDJ2		-1	1	ns
Data Bit Width Distortion after SOP	TFSOP	Note 8; Figure 7-52	-5	5	ns
Hub EOP Delay Relative to THDD	TFEOPD	Note 8; Figure 7-53	0	15	ns
Hub EOP Output Width Skew	TFHESK	Note 8; Figure 7-53	-15	15	ns
Low-speed Hub Characteristics (as measured at connectors):					
Driver Characteristics: (Refer to Table 7-10)		Downstream facing ports configured as low-speed			
Hub Differential Data Delay	TLHDD	Note 7, 8; Figure 7-52		300	ns
Hub Differential Driver Jitter (including cable):		Note 7, 8; Figure 7-52			
Downstream facing port :					
To Next Transition	TLDHJ1		-45	45	ns
For Paired Transitions	TLDHJ2		-15	15	ns
Upstream facing port:					
To Next Transition	TLUHJ1		-45	45	ns
For Paired Transitions	TLUHJ2		-45	45	ns
Data Bit Width Distortion after SOP	TLSOP	Note 8; Figure 7-52	-60	60	ns
Hub EOP Delay Relative to THDD	TLEOPD	Note 8; Figure 7-53	0	200	ns
Hub EOP Output Width Skew	TLHESK	Note 8; Figure 7-53	-300	+300	ns

Table 7-11. Hub/Repeater Electrical Characteristics (Continued)

Parameter	Symbol	Conditions	Min.	Max.	Units
High-speed Hub Characteristics (as measured at connectors):					
Driver Characteristics: (Refer to Table 7-8)		Upstream facing port and downstream facing ports configured as high-speed			
Hub Data Delay (without cable):	THSHDD	Section 7.1.14.2		36 high-speed bit times + 4 ns	
Hub Data Jitter:		Specified by eye patterns in Section 7.1.2.2			
Hub Delay Variation Range:	THSHDV	Section 7.1.14.2		5 high-speed bit times	

Table 7-12. Cable Characteristics (Note 14)

Parameter	Symbol	Conditions	Min	Max	Units
VBUS Voltage drop for detachable cables	VBUSD	Section 7.2.2		125	mV
GND Voltage drop (for all cables)	VGNDD	Section 7.2.2		125	mV
Differential Cable Impedance (full-/high-speed)	ZO	(90 Ω ±15%);	76.5	103.5	Ω
Common mode cable impedance (full-/high-speed)	ZCM	(30 Ω ±30%);	21.0	39.0	Ω
Cable Delay (one way) Full-/high-speed Low-speed	 TFSCBL TLSCBL	Section 7.1.16		 26 18	 ns ns
Cable Skew	TSKEW	Section 7.1.3		100	ps
Unmated Contact Capacitance	CUC	Section 6.7		2	pF
Cable loss		Specified by table and graph in Section 7.1.17			

Note 1: Measured at A plug.

Note 2: Measured at A receptacle.

Note 3: Measured at B receptacle.

Note 4: Measured at A or B connector.

Note 5: Measured with RL of 1.425 kΩ to 3.6 V.

Note 6: Measured with RL of 14.25 kΩ to GND.

Note 7: Timing difference between the differential data signals.

Note 8: Measured at crossover point of differential data signals.

Note 9: The maximum load specification is the maximum effective capacitive load allowed that meets the target VBUS drop of 330 mV.

Note 10: Excluding the first transition from the Idle state.

Note 11: The two transitions should be a (nominal) bit time apart.

Note 12: For both transitions of differential signaling.

Note 13: Must accept as valid EOP.

Note 14: Single-ended capacitance of D+ or D- is the capacitance of D+/D- to all other conductors and, if present, shield in the cable. That is, to measure the single-ended capacitance of D+, short D-, VBUS, GND, and the shield line together and measure the capacitance of D+ to the other conductors.

Note 15: For high power devices (non-hubs) when enabled for remote wakeup.

Table 7-13. Hub Event Timings

Event Description	Symbol	Conditions	Min	Max	Unit
Time to detect a downstream facing port connect event Awake Hub Suspended Hub	TDCNN	Section 11.5 and Section 7.1.7.3	 2.5 2.5	 2000 12000	 µs µs
Time to detect a disconnect event at a hub's downstream facing port	TDDIS	Section 7.1.7.3	2	2.5	µs
Duration of driving resume to a downstream port; only from a controlling hub	TDRSMDN	Nominal; Section 7.1.7.7 and Section 11.5	20		ms
Time from detecting downstream resume to rebroadcast	TURSM	Section 7.1.7.7		1.0	ms
Duration of driving reset to a downstream facing port	TDRST	Only for a SetPortFeature (PORT_RESET) request; Section 7.1.7.5 and Section 11.5	10	20	ms
Overall duration of driving reset to downstream facing port, root hub	TDRSTR	Only for root hubs; Section 7.1.7.5	50		ms
Maximum interval between reset segments used to create TDRSTR	TRHRSI	Only for root hubs; each reset pulse must be of length TDRST; Section 7.1.7.5		3	ms
Time to detect a long K from upstream	TURLK	Section 11.6	2.5	100	µs
Time to detect a long SE0 from upstream	TURLSE0	Section 11.6	2.5	10000	µs
Duration of repeating SE0 upstream (for low-/full-speed repeater)	TURPSE0	Section 11.6		23	FS bit times
Duration of sending SE0 upstream after EOF1 (for low-/full-speed repeater)	TUDEOP	*Optional* Section 11.6		2	FS bit times
Inter-packet Delay (for high-speed) for packets traveling in same direction	THSIPDSD	Section 7.1.18.2	88		bit times
Inter-packet Delay (for high-speed) for packets traveling in opposite direction	THSIPDOD	Section 7.1.18.2	8		bit times

Table 7-13. Hub Event Timings (Continued)

Event Description	Symbol	Conditions	Min	Max	Unit
Inter-packet delay for device/root hub response w/detachable cable for high-speed	THSRSPIPD1	Section 7.1.18.2		192	bit times
Reset Handshake Protocol:					
Time for which a Chirp J or Chirp K must be continuously detected (filtered) by hub or device during Reset handshake	TFILT	Section 7.1.7.5	2.5		µs
Time after end of device Chirp K by which hub must start driving first Chirp K in the hub's chirp sequence	TWTDCH	Section 7.1.7.5		100	µs
Time for which each individual Chirp J or Chirp K in the chirp sequence is driven downstream by hub during reset	TDCHBIT	Section 7.1.7.5	40	60	µs
Time before end of reset by which a hub must end its downstream chirp sequence	TDCHSE0	Section 7.1.7.5	100	500	µs

Table 7-14. Device Event Timings

Parameter	Symbol	Conditions	Min	Max	Units
Time from internal power good to device pulling D+/D- beyond VIHZ (min) (signaling attach)	TSIGATT	Figure 7-29		100	ms
Debounce interval provided by USB system software after attach	TATTDB	Figure 7-29		100	ms
Maximum time a device can draw power >suspend power when bus is continuously in idle state	T2SUSP	Section 7.1.7.6		10	ms
Maximum duration of suspend averaging interval	TSUSAVGI	Section 7.2.3		1	s
Period of idle bus before device can initiate resume	TWTRSM	Device must be remote-wakeup enabled Section 7.1.7.5	5		ms
Duration of driving resume upstream	TDRSMUP	Section 7.1.7.7	1	15	ms
Resume Recovery Time	TRSMRCY	Provided by USB System Software; Section 7.1.7.7	10		ms
Time to detect a reset from upstream for non high-speed capable devices	TDETRST	Section 7.1.7.5	2.5	10000	µs
Reset Recovery Time	TRSTRCY	Section 7.1.7.5		10	ms
Inter-packet Delay (for low-/full-speed)	TIPD	Section 7.1.18	2		bit times
Inter-packet delay for device response w/detachable cable for low-/full-speed	TRSPIPD1	Section 7.1.18		6.5	bit times
Inter-packet delay for device response w/captive cable for low-/full-speed	TRSPIPD2	Section 7.1.18		7.5	bit times

Table 7-14. Device Event Timings (Continued)

Parameter	Symbol	Conditions	Min	Max	Units
SetAddress() Completion Time	TDSETADDR	Section 9.2.6.3		50	ms
Time to complete standard request with no data	TDRQCMPLTND	Section 9.2.6.4		50	ms
Time to deliver first and subsequent (except last) data for standard request	TDRETDATA1	Section 9.2.6.4		500	ms
Time to deliver last data for standard request	TDRETDATAN	Section 9.2.6.4		50	ms
Inter-packet delay for device response w/captive cable (high-speed)	THSRSPIPD2	Section 7.1.18.2		192 bit times + 52 ns	
SetAddress() Completion Time	TDSETADDR	Section 9.2.6.3		50	ms
Time to complete standard request with no data	TDRQCMPLTND	Section 9.2.6.4		50	ms
Reset Handshake Protocol:					
Time for which a suspended high-speed capable device must see a continuous SE0 before beginning the high-speed detection handshake	TFILTSE0	Section 7.1.7.5	2.5		µs
Time a high-speed capable device operating in non-suspended full-speed must wait after start of SE0 before beginning the high-speed detection handshake	TWTRSTFS	Section 7.1.7.5	2.5	3000	µs
Time a high-speed capable device operating in high-speed must wait after start of SE0 before reverting to full-speed	TWTREV	Section 7.1.7.5	3.0	3.125	ms
Time a device must wait after reverting to full-speed before sampling the bus state for SE0 and beginning the high-speed detection handshake	TWTRSTHS	Section 7.1.7.5	100	875	µs

Table 7-14. Device Event Timings (Continued)

Parameter	Symbol	Conditions	Min	Max	Units
Minimum duration of a Chirp K from a high-speed capable device within the reset protocol	TUCH	Section 7.1.7.5	1.0		ms
Time after start of SE0 by which a high-speed capable device is required to have completed its Chirp K within the reset protocol	TUCHEND	Section 7.1.7.5		7.0	ms
Time between detection of downstream chirp and entering high-speed state	TWTHS	Section 7.1.7.5		500	µs
Time after end of upstream chirp at which device reverts to full-speed default state if no downstream chirp is detected	TWTFS	Section 7.1.7.5	1.0	2.5	ms

7.3.3 Timing Waveforms

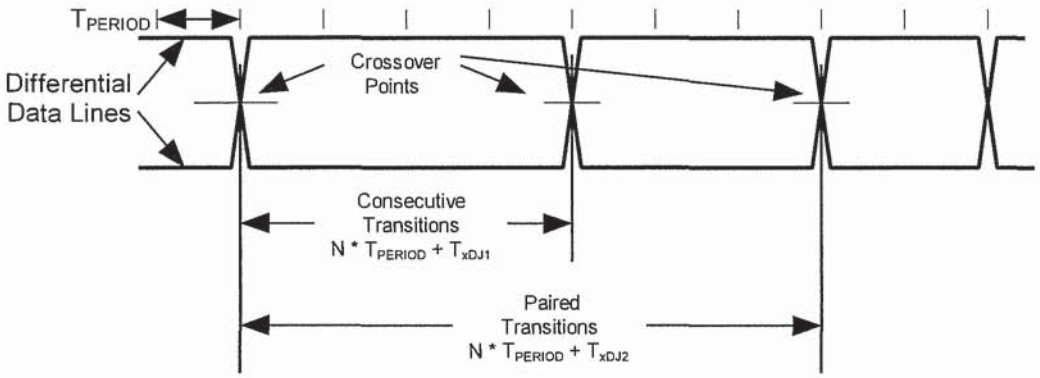


Figure 7-49. Differential Data Jitter for Low-/full-speed

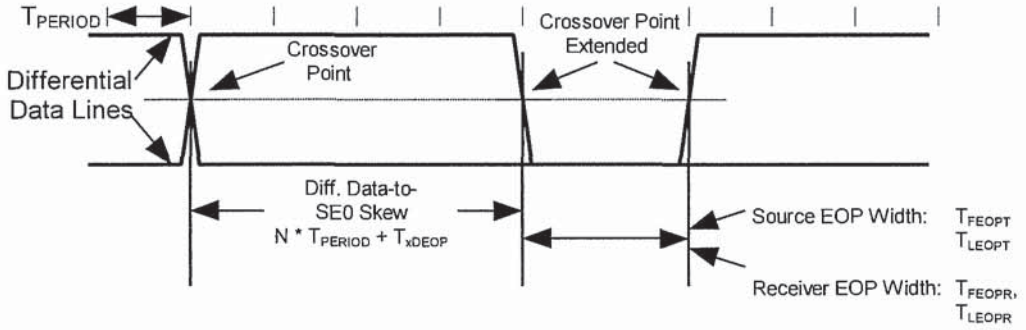


Figure 7-50. Differential-to-EOP Transition Skew and EOP Width for Low-/full-speed

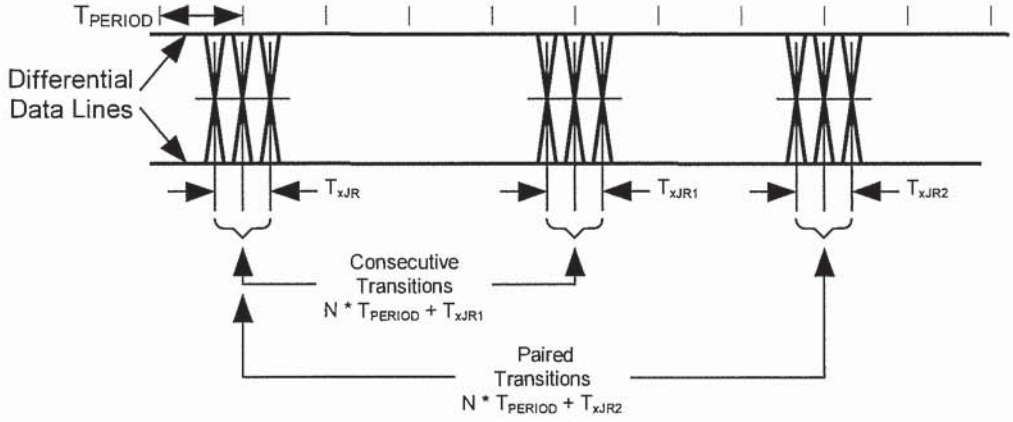
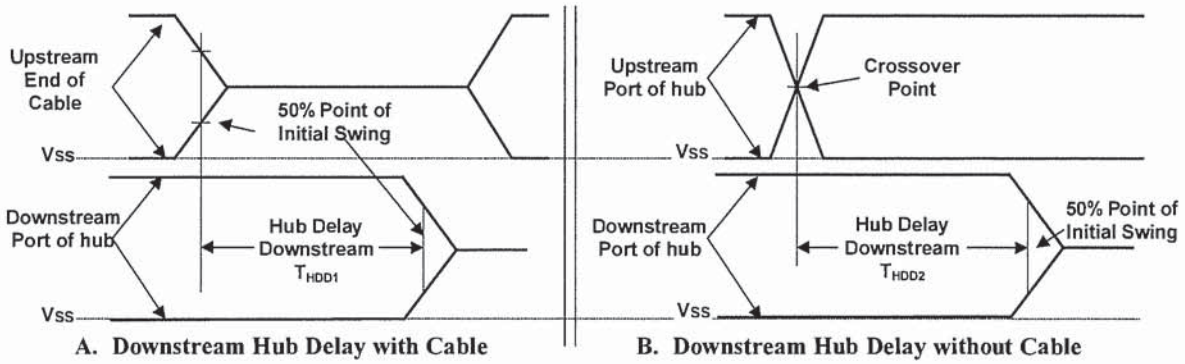


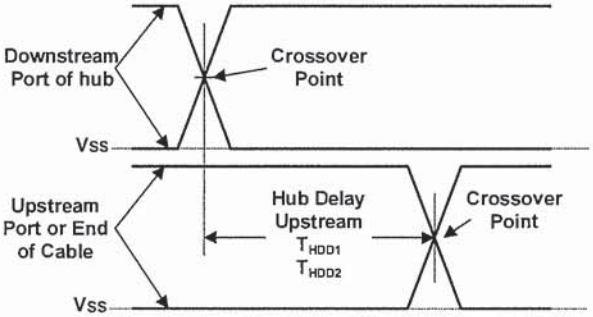
Figure 7-51. Receiver Jitter Tolerance for Low-/full-speed

TPERIOD is the data rate of the receiver that can have the range as defined in Section 7.1.11.



A. Downstream Hub Delay with Cable

B. Downstream Hub Delay without Cable



C. Upstream Hub Delay with or without Cable

Hub Differential Jitter:

$T_{HDJ1} = T_{HDDx}(J) - T_{HDDx}(K)$ or $T_{HDDx}(K) - T_{HDDx}(J)$ Consecutive Transitions

$T_{HDJ2} = T_{HDDx}(J) - T_{HDDx}(J)$ or $T_{HDDx}(K) - T_{HDDx}(K)$ Paired Transitions

Bit after SOP Width Distortion (same as data jitter for SOP and next J transition):

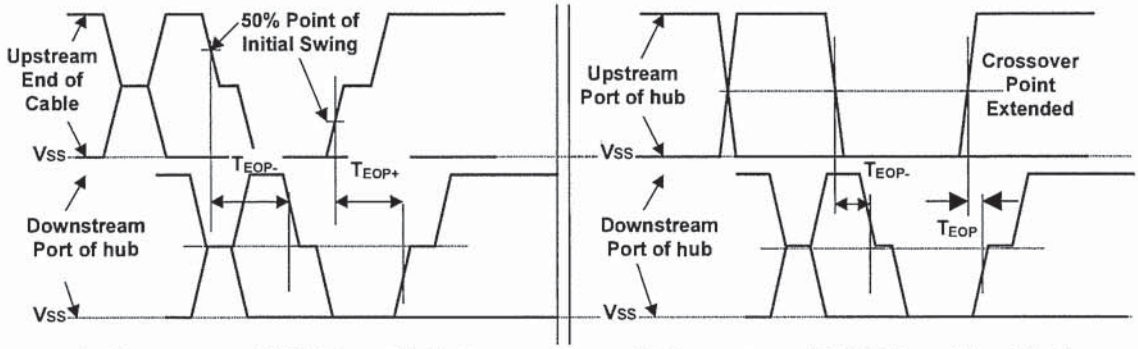
$T_{FSOP} = T_{HDDx}$(next J) - T_{HDDx}(SOP)

Low-speed timings are determined in the same way for:

T_{LHDD}, T_{LDHJ1}, T_{LDJH2}, T_{LUHJ1}, T_{LUJH2}, and T_{LSOP}

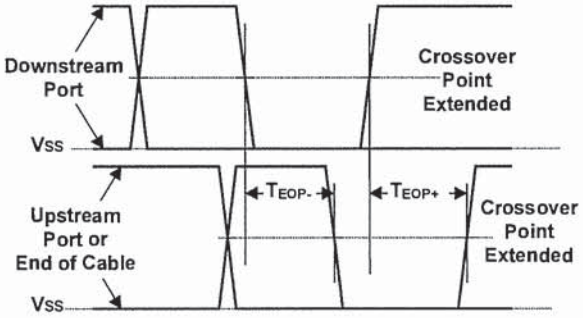
Figure 7-52. Hub Differential Delay, Differential Jitter, and SOP Distortion for Low-/full-speed

Measurement locations referenced in Figure 7-52 and Figure 7-53 are specified in Figure 7-38.



A. Downstream EOP Delay with Cable

B. Downstream EOP Delay without Cable



C. Upstream EOP Delay with or Without Cable

EOP Delay:

$T_{FEOPD} = T_{EOPy} - T_{HDDx}$

(T_{EOPy} means that this equation applies to T_{EOP-} and T_{EOP+})

EOP Skew:

$T_{FHESK} = T_{EOP+} - T_{EOP-}$

Low-speed timings are determined in the same way for:

T_{LEOPD} and T_{LHESK}

Figure 7-53. Hub EOP Delay and EOP Skew for Low-/full-speed

Chapter 8
Protocol Layer

This chapter presents a bottom-up view of the USB protocol, starting with field and packet definitions. This is followed by a description of packet transaction formats for different transaction types. Link layer flow control and transaction level fault recovery are then covered. The chapter finishes with a discussion of retry synchronization, babble, loss of bus activity recovery, and high-speed PING protocol.

8.1 Byte/Bit Ordering

Bits are sent out onto the bus least-significant bit (LSb) first, followed by the next LSb, through to the most-significant bit (MSb) last. In the following diagrams, packets are displayed such that both individual bits and fields are represented (in a left to right reading order) as they would move across the bus.

Multiple byte fields in standard descriptors, requests, and responses are interpreted as and moved over the bus in little-endian order, i.e., LSB to MSB.

8.2 SYNC Field

All packets begin with a synchronization (SYNC) field, which is a coded sequence that generates a maximum edge transition density. It is used by the input circuitry to align incoming data with the local clock. A SYNC from an initial transmitter is defined to be eight bits in length for full/low-speed and 32 bits for high-speed. Received SYNC fields may be shorter as described in Chapter 7. SYNC serves only as a synchronization mechanism and is not shown in the following packet diagrams (refer to Section 7.1.10). The last two bits in the SYNC field are a marker that is used to identify the end of the SYNC field and, by inference, the start of the PID.

8.3 Packet Field Formats

Field formats for the token, data, and handshake packets are described in the following section. Packet bit definitions are displayed in unencoded data format. The effects of NRZI coding and bit stuffing have been removed for the sake of clarity. All packets have distinct Start- and End-of-Packet delimiters. The Start-of-Packet (SOP) delimiter is part of the SYNC field, and the End-of-Packet (EOP) delimiter is described in Chapter 7.

8.3.1 Packet Identifier Field

A packet identifier (PID) immediately follows the SYNC field of every USB packet. A PID consists of a four-bit packet type field followed by a four-bit check field as shown in Figure 8-1. The PID indicates the type of packet and, by inference, the format of the packet and the type of error detection applied to the packet. The four-bit check field of the PID ensures reliable decoding of the PID so that the remainder of the packet is interpreted correctly. The PID check field is generated by performing a one's complement of the packet type field. A PID error exists if the four PID check bits are not complements of their respective packet identifier bits.

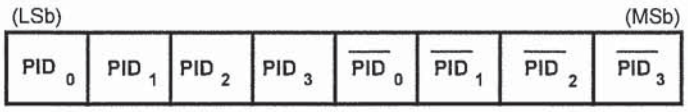


Figure 8-1. PID Format

The host and all functions must perform a complete decoding of all received PID fields. Any PID received with a failed check field or which decodes to a non-defined value is assumed to be corrupted and it, as well

as the remainder of the packet, is ignored by the packet receiver. If a function receives an otherwise valid PID for a transaction type or direction that it does not support, the function must not respond. For example, an IN-only endpoint must ignore an OUT token. PID types, codings, and descriptions are listed in Table 8-1.

Table 8-1. PID Types

PID Type	PID Name	PID<3:0>*	Description
Token	OUT	0001B	Address + endpoint number in host-to-function transaction
	IN	1001B	Address + endpoint number in function-to-host transaction
	SOF	0101B	Start-of-Frame marker and frame number
	SETUP	1101B	Address + endpoint number in host-to-function transaction for SETUP to a control pipe
Data	DATA0	0011B	Data packet PID even
	DATA1	1011B	Data packet PID odd
	DATA2	0111B	Data packet PID high-speed, high bandwidth isochronous transaction in a microframe (see Section 5.9.2 for more information)
	MDATA	1111B	Data packet PID high-speed for split and high bandwidth isochronous transactions (see Sections 5.9.2, 11.20, and 11.21 for more information)
Handshake	ACK	0010B	Receiver accepts error-free data packet
	NAK	1010B	Receiving device cannot accept data or transmitting device cannot send data
	STALL	1110B	Endpoint is halted or a control pipe request is not supported
	NYET	0110B	No response yet from receiver (see Sections 8.5.1 and 11.17-11.21)
Special	PRE	1100B	(Token) Host-issued preamble. Enables downstream bus traffic to low-speed devices.
	ERR	1100B	(Handshake) Split Transaction Error Handshake (reuses PRE value)
	SPLIT	1000B	(Token) High-speed Split Transaction Token (see Section 8.4.2)
	PING	0100B	(Token) High-speed flow control probe for a bulk/control endpoint (see Section 8.5.1)
	Reserved	0000B	Reserved PID

*Note: PID bits are shown in MSb order. When sent on the USB, the rightmost bit (bit 0) will be sent first.

PIDs are divided into four coding groups: token, data, handshake, and special, with the first two transmitted PID bits (PID<0:1>) indicating which group. This accounts for the distribution of PID codes.

8.3.2 Address Fields

Function endpoints are addressed using two fields: the function address field and the endpoint field. A function needs to fully decode both address and endpoint fields. Address or endpoint aliasing is not permitted, and a mismatch on either field must cause the token to be ignored. Accesses to non-initialized endpoints will also cause the token to be ignored.

8.3.2.1 Address Field

The function address (ADDR) field specifies the function, via its address, that is either the source or destination of a data packet, depending on the value of the token PID. As shown in Figure 8-2, a total of 128 addresses are specified as ADDR<6:0>. The ADDR field is specified for IN, SETUP, and OUT tokens and the PING and SPLIT special token. By definition, each ADDR value defines a single function. Upon reset and power-up, a function's address defaults to a value of zero and must be programmed by the host during the enumeration process. Function address zero is reserved as the default address and may not be assigned to any other use.

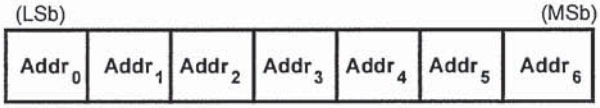


Figure 8-2. ADDR Field

8.3.2.2 Endpoint Field

An additional four-bit endpoint (ENDP) field, shown in Figure 8-3, permits more flexible addressing of functions in which more than one endpoint is required. Except for endpoint address zero, endpoint numbers are function-specific. The endpoint field is defined for IN, SETUP, and OUT tokens and the PING special token. All functions must support a control pipe at endpoint number zero (the Default Control Pipe). Low-speed devices support a maximum of three pipes per function: a control pipe at endpoint number zero plus two additional pipes (either two control pipes, a control pipe and a interrupt endpoint, or two interrupt endpoints). Full-speed and high-speed functions may support up to a maximum of 16 IN and OUT endpoints.

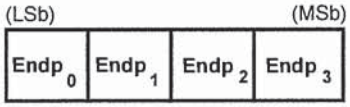


Figure 8-3. Endpoint Field

8.3.3 Frame Number Field

The frame number field is an 11-bit field that is incremented by the host on a per-frame basis. The frame number field rolls over upon reaching its maximum value of 7FFH and is sent only in SOF tokens at the start of each (micro)frame.

8.3.4 Data Field

The data field may range from zero to 1,024 bytes and must be an integral number of bytes. Figure 8-4 shows the format for multiple bytes. Data bits within each byte are shifted out LSb first.

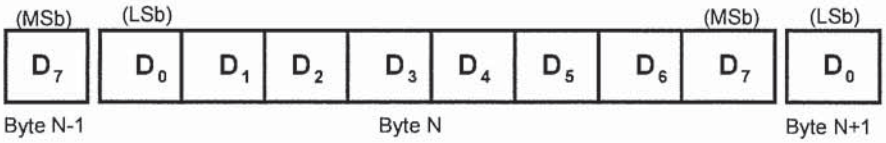


Figure 8-4. Data Field Format

Data packet size varies with the transfer type, as described in Chapter 5.

8.3.5 Cyclic Redundancy Checks

Cyclic redundancy checks (CRCs) are used to protect all non-PID fields in token and data packets. In this context, these fields are considered to be protected fields. The PID is not included in the CRC check of a packet containing a CRC. All CRCs are generated over their respective fields in the transmitter before bit stuffing is performed. Similarly, CRCs are decoded in the receiver after stuffed bits have been removed. Token and data packet CRCs provide 100% coverage for all single- and double-bit errors. A failed CRC is considered to indicate that one or more of the protected fields is corrupted and causes the receiver to ignore those fields and, in most cases, the entire packet.

For CRC generation and checking, the shift registers in the generator and checker are seeded with an all-ones pattern. For each data bit sent or received, the high order bit of the current remainder is XORed with the data bit and then the remainder is shifted left one bit and the low-order bit set to zero. If the result of that XOR is one, then the remainder is XORed with the generator polynomial.

When the last bit of the checked field is sent, the CRC in the generator is inverted and sent to the checker MSb first. When the last bit of the CRC is received by the checker and no errors have occurred, the remainder will be equal to the polynomial residual.

A CRC error exists if the computed checksum remainder at the end of a packet reception does not match the residual.

Bit stuffing requirements must be met for the CRC, and this includes the need to insert a zero at the end of a CRC if the preceding six bits were all ones.

8.3.5.1 Token CRCs

A five-bit CRC field is provided for tokens and covers the ADDR and ENDP fields of IN, SETUP, and OUT tokens or the time stamp field of an SOF token. The PING and SPLIT special tokens also include a five-bit CRC field. The generator polynomial is:

$$G(X) = X^5 + X^2 + 1$$

The binary bit pattern that represents this polynomial is 00101B. If all token bits are received without error, the five-bit residual at the receiver will be 01100B.

8.3.5.2 Data CRCs

The data CRC is a 16-bit polynomial applied over the data field of a data packet. The generating polynomial is:

$$G(X) = X^{16} + X^{15} + X^2 + 1$$

The binary bit pattern that represents this polynomial is 1000000000000101B. If all data and CRC bits are received without error, the 16-bit residual will be 1000000000001101B.

8.4 Packet Formats

This section shows packet formats for token, data, and handshake packets. Fields within a packet are displayed in these figures in the order in which bits are shifted out onto the bus.

8.4.1 Token Packets

Figure 8-5 shows the field formats for a token packet. A token consists of a PID, specifying either IN, OUT, or SETUP packet type and ADDR and ENDP fields. The PING special token packet also has the same fields as a token packet. For OUT and SETUP transactions, the address and endpoint fields uniquely identify the endpoint that will receive the subsequent Data packet. For IN transactions, these fields uniquely identify which endpoint should transmit a Data packet. For PING transactions, these fields uniquely identify which endpoint will respond with a handshake packet. Only the host can issue token packets. An IN PID defines a Data transaction from a function to the host. OUT and SETUP PIDs define Data transactions from the host to a function. A PING PID defines a handshake transaction from the function to the host.

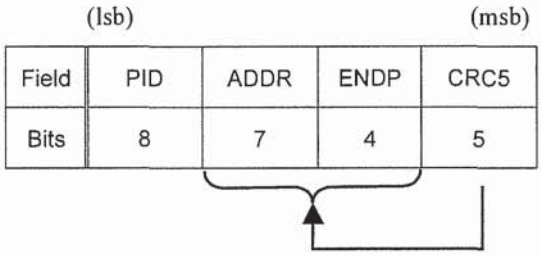


Figure 8-5. Token Format

Token packets have a five-bit CRC that covers the address and endpoint fields as shown above. The CRC does not cover the PID, which has its own check field. Token and SOF packets are delimited by an EOP after three bytes of packet field data. If a packet decodes as an otherwise valid token or SOF but does not terminate with an EOP after three bytes, it must be considered invalid and ignored by the receiver.

8.4.2 Split Transaction Special Token Packets

USB defines a special token for split transactions: SPLIT. This is a 4 byte token packet compared to other normal 3 byte token packets. The split transaction token packet provides additional transaction types with additional transaction specific information. The split transaction token is used to support split transactions between the host controller communicating with a hub operating at high speed with full-/low-speed devices to some of its downstream facing ports. There are two split transactions defined that use the SPLIT special token: a start-split transaction (SSPLIT) and a complete-split transaction (CSPLIT). A field in the SPLIT special token, described in the following sections, indicates the specific split transaction.

8.4.2.1 Split Transactions

A high-speed split transaction is used only between the host controller and a hub when the hub has full-/low-speed devices attached to it. This high-speed split transaction is used to initiate a full-/low-speed transaction via the hub and some full-/low-speed device endpoint. The high-speed split transaction also allows the completion status of the full-/low-speed transaction to be retrieved from the hub. This approach allows the host controller to start a full-/low-speed transaction via a high-speed transaction and then continue with other high-speed transactions without having to wait for the full-/low-speed transaction to proceed/complete at the slower speed. See Chapter 11 for more details about the state machines and transaction definitions of split transactions.

A high-speed split transaction has two parts: a start-split and a complete-split. Split transactions are only defined to be used between the host controller and a hub. No other high-speed or full-/low-speed devices ever use split transactions.

Figure 8-6 shows the packets composing a generic start-split transaction. There are two packets in the token phase: the SPLIT special token and a full-/low-speed token. Depending on the direction of data transfer and whether a handshake is defined for the transaction type, the token phase is optionally followed by a data packet and a handshake packet. Start split transactions can consist of 2, 3, or 4 packets as determined by the specific transfer type and data direction.

Figure 8-6. Packets in a Start-split Transaction

Figure 8-7 shows the packets composing a generic complete-split transaction. There are two packets in the token phase: the SPLIT special token and a full-/low-speed token. A data or handshake packet follows the token phase packets in the complete-split depending on the data transfer direction and specific transaction type. Complete split transactions can consist of 2 or 3 packets as determined by the specific transfer type and data direction.

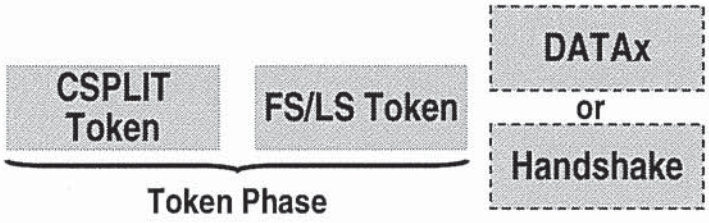


Figure 8-7. Packets in a Complete-split Transaction

The results of a split transaction are returned by a complete-split transaction. Figure 8-8 shows this conceptual "conversion" for an example interrupt IN transfer type. The host issues a start-split (indicated with 1) to the hub and then can proceed with other high-speed transactions. The start-split causes the hub to issue a full-/low-speed IN token sometime later (indicated by 2). The device responds to the IN token (in this example) with a data packet and the hub responds with a handshake to the device. Finally, the host sometime later issues a complete-split (indicated by 3) to retrieve the data provided by the device. Note that in the example, the hub provided the full-/low-speed handshake (ACK in this example) to the device endpoint before the complete-split, and the complete-split did not provide a high-speed handshake to the hub.

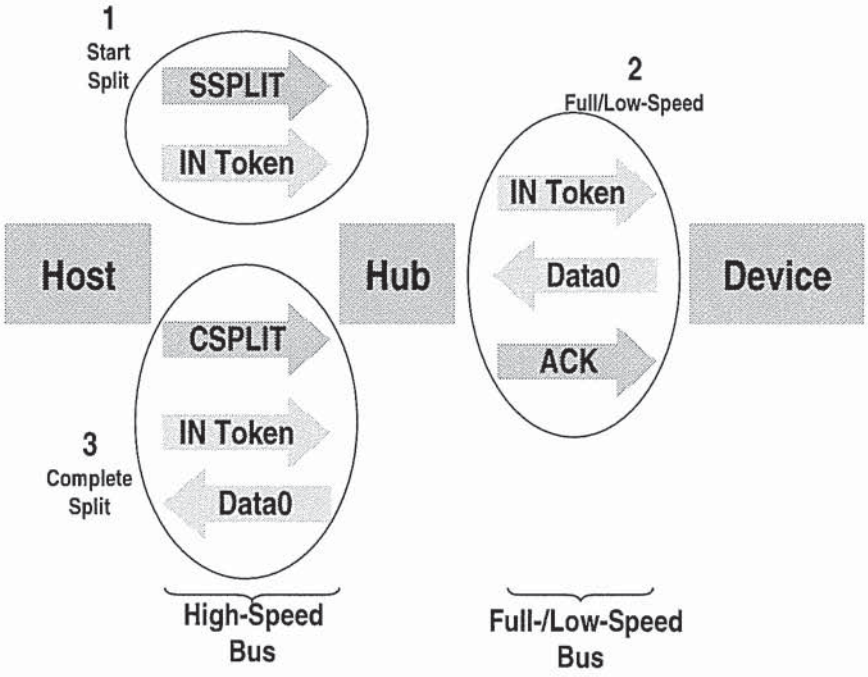


Figure 8-8. Relationship of Interrupt IN Transaction to High-speed Split Transaction

A normal full-/low-speed OUT transaction is similarly conceptually "converted" into start-split and complete-split transactions. Figure 8-9 shows this "conversion" for an example interrupt OUT transfer type. The host issues a start-split transaction consisting of a SSPLIT special token, an OUT token, and a DATA packet. The hub sometime later issues the OUT token and DATA packet on the full-/low-speed bus. The device responds with a handshake. Sometime later, the host issues the complete-split transaction and the hub responds with the results (either full-/low-speed data or handshake) provided by the device.

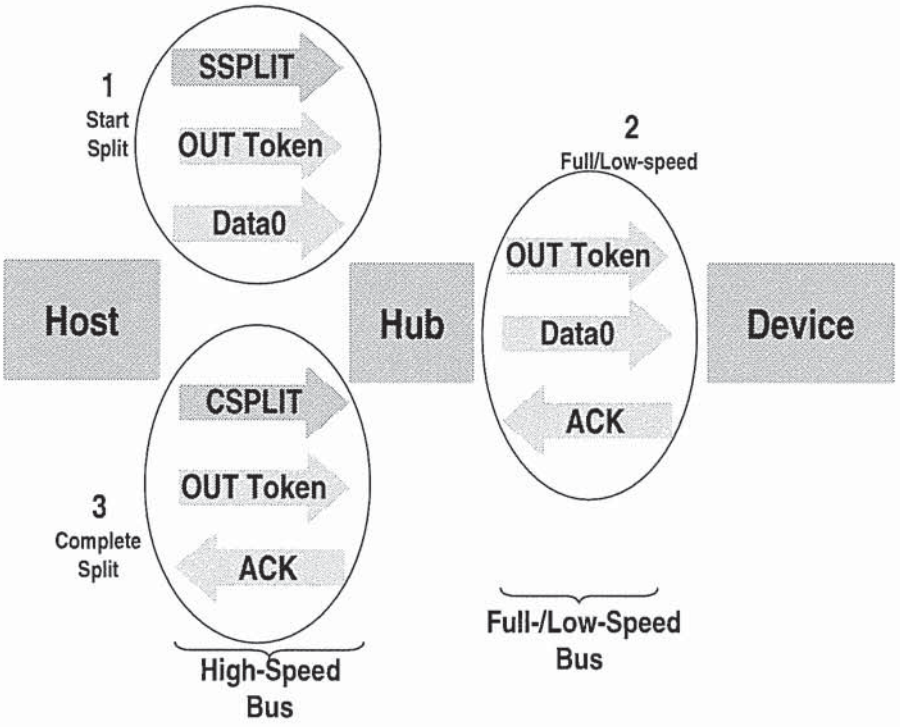


Figure 8-9. Relationship of Interrupt OUT Transaction to High-speed Split OUT Transaction

The next two sections describe the fields composing the detailed start- and complete-split token packets. Figure 8-10 and Figure 8-12 show the fields in the split-transaction token packet. The SPLIT special token follows the general token format and starts with a PID field (after a SYNC) and ends with a CRC5 field (and EOP). Start-split and complete-split token packets are both 4 bytes long. SPLIT transactions must only originate from the host. The start-split token is defined in Section 8.4.2.2 and the complete-split token is defined in Section 8.4.2.3.

8.4.2.2 Start-Split Transaction Token

(lsb) (msb)

Field	SPLIT PID	Hub Addr	SC	Port	S	E	ET	CRC5
Bits	8	7	1	7	1	1	2	5

Figure 8-10. Start-split (SSPLIT) Token

The Hub addr field contains the USB device address of the hub supporting the specified full-/low-speed device for this full-/low-speed transaction. This field has the same definition as the ADDR field definition in Section 8.3.2.1.

A SPLIT special token packet with the SC (Start/Complete) field set to zero indicates that this is a start-split transaction (SSPLIT).

The Port field contains the port number of the target hub for which this full-/low-speed transaction is destined. As shown in Figure 8-11, a total of 128 ports are specified as PORT<6:0>. The host must correctly set the port field for single and multiple TT hub implementations. A single TT hub implementation may ignore the port field.

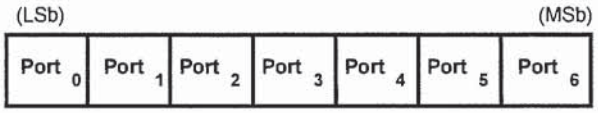


Figure 8-11. Port Field

The S (Speed) field specifies the speed for this interrupt or control transaction as follows:

- 0 – Full speed
- 1 – Low speed

For bulk IN/OUT and isochronous IN start-splits, the S field must be set to zero. For bulk/control IN/OUT, interrupt IN/OUT, and isochronous IN start-splits, the E field must be set to zero.

For full-speed isochronous OUT start-splits, the S[1] (Start) and E (End) fields specify how the high-speed data payload corresponds to data for a full-speed data packet as shown in Table 8-2.

Table 8-2. Isochronous OUT Payload Continuation Encoding

S	E	High-speed to Full-speed Data Relation
0	0	High-speed data is the middle of the full-speed data payload
0	1	High-speed data is the end of the full-speed data payload
1	0	High-speed data is the beginning of the full-speed data payload
1	1	High-speed data is all of the full-speed data payload.

Isochronous OUT start-split transactions use these encodings to allow the hub to detect various error cases due to lack of receiving start-split transactions for an endpoint with a data payload that requires multiple start-splits. For example, a large full-speed data payload may require three start-split transactions: a start-split/beginning, a start-split/middle and a start-split/end. If any of these transactions is not received by the hub, it will either ignore the full-speed transaction (if the start-split/beginning is not received), or it will force an error for the corresponding full-speed transaction (if one of the other two transactions are not received). Other error conditions can be detected by not receiving a start-split during a microframe.

The ET (Endpoint Type) field specifies the endpoint type of the full-/low-speed transaction as shown in Table 8-3.

[1] The S bit can be reused for these encodings since isochronous transactions must not be low speed.

Table 8-3. Endpoint Type Values in Split Special Token

ET value (msb:lsb)	Endpoint Type
00	Control
01	Isochronous
10	Bulk
11	Interrupt

This field tells the hub which split transaction state machine to use for this full-/low-speed transaction.

The full-/low-speed device address and endpoint number information is contained in the normal token packet that follows the SPLIT special token packet.

8.4.2.3 Complete-Split Transaction Token

(lsb) (msb)

Field	SPLIT PID	Hub Addr	SC	Port	S	U	ET	CRC5
Bits	8	7	1	7	1	1	2	5

Figure 8-12. Complete-split (CSPLIT) Transaction Token

A SPLIT special token packet with the SC field set to one indicates that this is a complete-split transaction (CSPLIT).

The U bit is reserved/unused and must be reset to zero(0B).

The other fields of the complete-split token packet have the same definitions as for the start-split token packet.

8.4.3 Start-of-Frame Packets

Start-of-Frame (SOF) packets are issued by the host at a nominal rate of once every 1.00 ms ±0.0005 ms for a full-speed bus and 125 µs ±0.0625 µs for a high-speed bus. SOF packets consist of a PID indicating packet type followed by an 11-bit frame number field as illustrated in Figure 8-13.

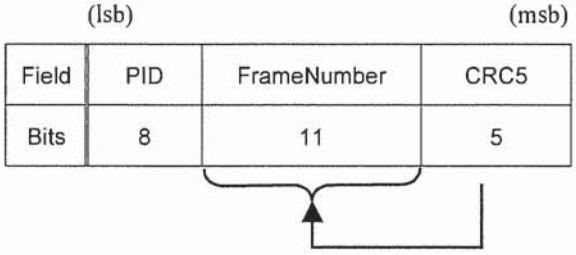


Figure 8-13. SOF Packet

The SOF token comprises the token-only transaction that distributes an SOF marker and accompanying frame number at precisely timed intervals corresponding to the start of each frame. All high-speed and full-speed functions, including hubs, receive the SOF packet. The SOF token does not cause any receiving function to generate a return packet; therefore, SOF delivery to any given function cannot be guaranteed.

The SOF packet delivers two pieces of timing information. A function is informed that an SOF has occurred when it detects the SOF PID. Frame timing sensitive functions, that do not need to keep track of frame number (e.g., a full-speed operating hub), need only decode the SOF PID; they can ignore the frame number and its CRC. If a function needs to track frame number, it must comprehend both the PID and the time stamp. Full-speed devices that have no particular need for bus timing information may ignore the SOF packet.

8.4.3.1 USB Frames and Microframes

USB defines a full-speed 1 ms frame time indicated by a Start Of Frame (SOF) packet each and every 1ms period with defined jitter tolerances. USB also defines a high-speed microframe with a 125 µs frame time with related jitter tolerances (See Chapter 7). SOF packets are generated (by the host controller or hub transaction translator) every 1ms for full-speed links. SOF packets are also generated after the next seven 125 µs periods for high-speed links.

Figure 8-14 shows the relationship between microframes and frames.

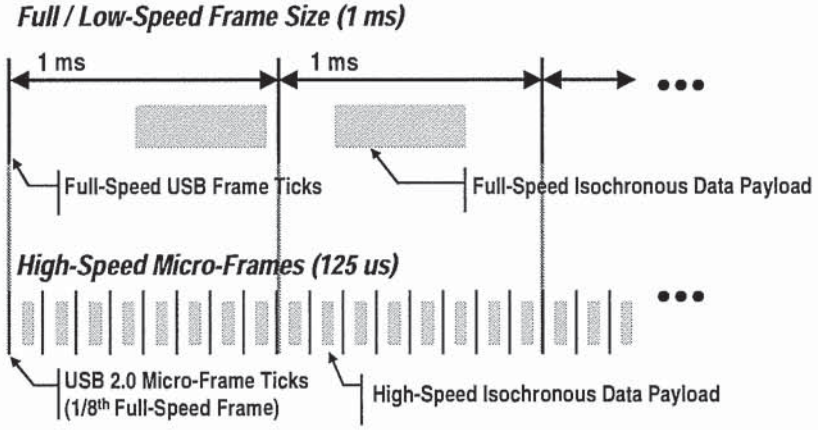


Figure 8-14. Relationship between Frames and Microframes

High-speed devices see an SOF packet with the same frame number eight times (every 125 µs) during each 1 ms period. If desired, a high-speed device can locally determine a particular microframe "number" by detecting the SOF that had a different frame number than the previous SOF and treating that as the zeroth microframe. The next seven SOFs with the same frame number can be treated as microframes 1 through 7.

8.4.4 Data Packets

A data packet consists of a PID, a data field containing zero or more bytes of data, and a CRC as shown in Figure 8-15. There are four types of data packets, identified by differing PIDs: DATA0, DATA1, DATA2 and MDATA. Two data packet PIDs (DATA0 and DATA1) are defined to support data toggle synchronization (refer to Section 8.6). All four data PIDs are used in data PID sequencing for high bandwidth high-speed isochronous endpoints (refer to Section 5.9). Three data PIDs (MDATA, DATA0, DATA1) are used in split transactions (refer to Sections 11.17-11.21).

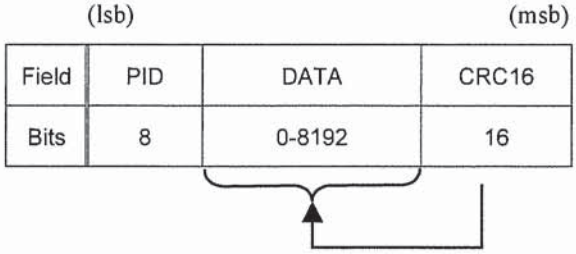


Figure 8-15. Data Packet Format

Data must always be sent in integral numbers of bytes. The data CRC is computed over only the data field in the packet and does not include the PID, which has its own check field.

The maximum data payload size allowed for low-speed devices is 8 bytes. The maximum data payload size for full-speed devices is 1023. The maximum data payload size for high-speed devices is 1024 bytes.

8.4.5 Handshake Packets

Handshake packets, as shown in Figure 8-16, consist of only a PID. Handshake packets are used to report the status of a data transaction and can return values indicating successful reception of data, command acceptance or rejection, flow control, and halt conditions. Only transaction types that support flow control can return handshakes. Handshakes are always returned in the handshake phase of a transaction and may be returned, instead of data, in the data phase. Handshake packets are delimited by an EOP after one byte of packet field. If a packet decodes as an otherwise valid handshake but does not terminate with an EOP after one byte, it must be considered invalid and ignored by the receiver.

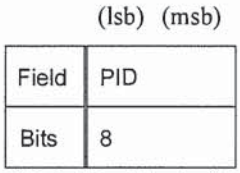


Figure 8-16. Handshake Packet

There are four types of handshake packets and one special handshake packet:

- **ACK** indicates that the data packet was received without bit stuff or CRC errors over the data field and that the data PID was received correctly. ACK may be issued either when sequence bits match and the receiver can accept data or when sequence bits mismatch and the sender and receiver must resynchronize to each other (refer to Section 8.6 for details). An ACK handshake is applicable only in transactions in which data has been transmitted and where a handshake is expected. ACK can be returned by the host for IN transactions and by a function for OUT, SETUP, or PING transactions.
- **NAK** indicates that a function was unable to accept data from the host (OUT) or that a function has no data to transmit to the host (IN). NAK can only be returned by functions in the data phase of IN transactions or the handshake phase of OUT or PING transactions. The host can never issue NAK.

NAK is used for flow control purposes to indicate that a function is temporarily unable to transmit or receive data, but will eventually be able to do so without need of host intervention.

- **STALL** is returned by a function in response to an IN token or after the data phase of an OUT or in response to a PING transaction (see Figure 8-30 and Figure 8-38). STALL indicates that a function is unable to transmit or receive data, or that a control pipe request is not supported. The state of a function after returning a STALL (for any endpoint except the default endpoint) is undefined. The host is not permitted to return a STALL under any condition.

  The STALL handshake is used by a device in one of two distinct occasions. The first case, known as "functional stall," is when the *Halt* feature associated with the endpoint is set. (The *Halt* feature is specified in Chapter 9 of this document.) A special case of the functional stall is the "commanded stall." Commanded stall occurs when the host explicitly sets the endpoint's *Halt* feature, as detailed in Chapter 9. Once a function's endpoint is halted, the function must continue returning STALL until the condition causing the halt has been cleared through host intervention.

  The second case, known as "protocol stall," is detailed in Section 8.5.3. Protocol stall is unique to control pipes. Protocol stall differs from functional stall in meaning and duration. A protocol STALL is returned during the Data or Status stage of a control transfer, and the STALL condition terminates at the beginning of the next control transfer (Setup). The remainder of this section refers to the general case of a functional stall.

- **NYET** is a high-speed only handshake that is returned in two circumstances. It is returned by a high-speed endpoint as part of the PING protocol described later in this chapter. NYET may also be returned by a hub in response to a split-transaction when the full-/low-speed transaction has not yet been completed or the hub is otherwise not able to handle the split-transaction. See Chapter 11 for more details.

- **ERR** is a high-speed only handshake that is returned to allow a high-speed hub to report an error on a full-/low-speed bus. It is only returned by a high-speed hub as part of the split transaction protocol. See Chapter 11 for more details.

8.4.6 Handshake Responses

Transmitting and receiving functions must return handshakes based upon an order of precedence detailed in Table 8-4 through Table 8-6. Not all handshakes are allowed, depending on the transaction type and whether the handshake is being issued by a function or the host. Note that if an error occurs during the transmission of the token to the function, the function will not respond with any packets until the next token is received and successfully decoded.

8.4.6.1 Function Response to IN Transactions

Table 8-4 shows the possible responses a function may make in response to an IN token. If the function is unable to send data, due to a halt or a flow control condition, it issues a STALL or NAK handshake, respectively. If the function is able to issue data, it does so. If the received token is corrupted, the function returns no response.

Table 8-4. Function Responses to IN Transactions

Token Received Corrupted	Function Tx Endpoint Halt Feature	Function Can Transmit Data	Action Taken
Yes	Don't care	Don't care	Return no response
No	Set	Don't care	Issue STALL handshake
No	Not set	No	Issue NAK handshake
No	Not set	Yes	Issue data packet

8.4.6.2 Host Response to IN Transactions

Table 8-5 shows the host response to an IN transaction. The host is able to return only one type of handshake: ACK. If the host receives a corrupted data packet, it discards the data and issues no response. If the host cannot accept data from a function, (due to problems such as internal buffer overrun) this condition is considered to be an error and the host returns no response. If the host is able to accept data and the data packet is received error-free, the host accepts the data and issues an ACK handshake.

Table 8-5. Host Responses to IN Transactions

Data Packet Corrupted	Host Can Accept Data	Handshake Returned by Host
Yes	N/A	Discard data, return no response
No	No	Discard data, return no response
No	Yes	Accept data, issue ACK

8.4.6.3 Function Response to an OUT Transaction

Handshake responses for an OUT transaction are shown in Table 8-6. Assuming successful token decode, a function, upon receiving a data packet, may return any one of the three handshake types. If the data packet was corrupted, the function returns no handshake. If the data packet was received error-free and the function's receiving endpoint is halted, the function returns STALL. If the transaction is maintaining sequence bit synchronization and a mismatch is detected (refer to Section 8.6 for details), then the function returns ACK and discards the data. If the function can accept the data and has received the data error-free, it returns ACK. If the function cannot accept the data packet due to flow control reasons, it returns NAK.

Table 8-6. Function Responses to OUT Transactions in Order of Precedence

Data Packet Corrupted	Receiver Halt Feature	Sequence Bits Match	Function Can Accept Data	Handshake Returned by Function
Yes	N/A	N/A	N/A	None
No	Set	N/A	N/A	STALL
No	Not set	No	N/A	ACK
No	Not set	Yes	Yes	ACK
No	Not set	Yes	No	NAK

8.4.6.4 Function Response to a SETUP Transaction

SETUP defines a special type of host-to-function data transaction that permits the host to initialize an endpoint's synchronization bits to those of the host. Upon receiving a SETUP token, a function must accept the data. A function may not respond to a SETUP token with either STALL or NAK, and the receiving function must accept the data packet that follows the SETUP token. If a non-control endpoint receives a SETUP token, it must ignore the transaction and return no response.

8.5 Transaction Packet Sequences

The packets that comprise a transaction varies depending on the endpoint type. There are four endpoint types: bulk, control, interrupt, and isochronous.

A host controller and device each require different state machines to correctly sequence each type of transaction. Figures in the following sections show state machines that define the correct sequencing of packets within a transaction of each type. The diagrams should not be taken as a required implementation, but to specify the required behavior.

Figure 8-17 shows the legend for the state machine diagrams. A circle with a three-line border indicates a reference to another (hierarchical) state machine. A circle with a two-line border indicates an initial state. A circle with a single-line border represents a simple state.

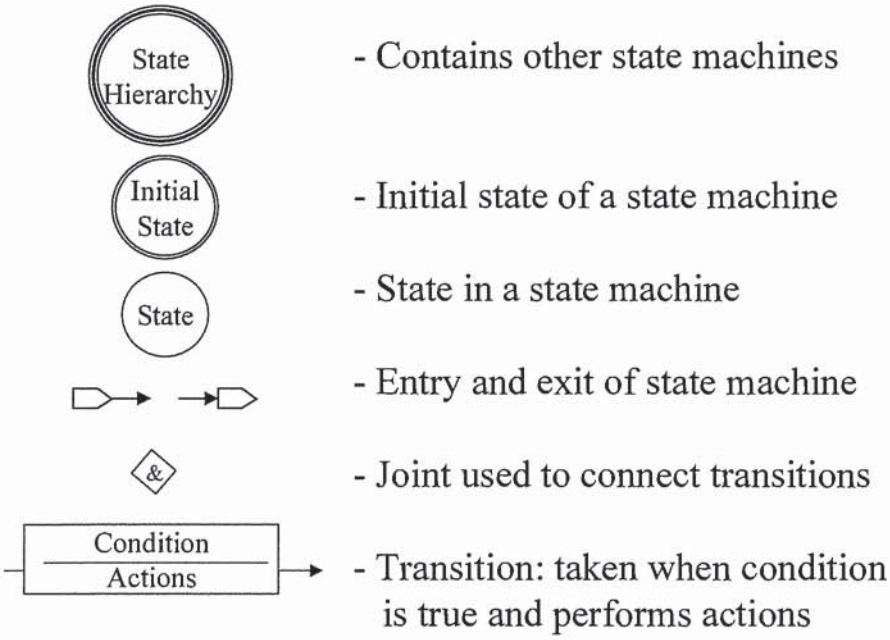


Figure 8-17. Legend for State Machines

The "tab" shapes with arrows are the entry or exit (respectively in the legend) to/from the state machine. The entry/exit relates to another state in a state machine at a higher level in the state machine hierarchy.

A diamond (joint) is used to join several transitions to a common point. A joint allows a single input transition with multiple output transitions or multiple input transitions and a single output transition. All conditions on the transitions of a path involving a joint must be true for the path to be taken. A path is simply a sequence of transitions involving one or more joints.

A transition is labeled with a block with a line in the middle separating the (upper) condition and the (lower) actions. The condition is required to be true to take the transition. The syntax for actions and conditions is VHDL. The actions are performed if the transition is taken. A circle includes a name in bold and optionally one or more actions that are performed upon entry to the state.

The host controller and device state machines are in a context as shown in Figure 8-18. The host controller determines the next transaction to run for an endpoint and issues a command (HC_cmd) to the host controller state machines. This causes the host controller state machines to issue one or more packets to move over the downstream bus (HSD1).

The device receives these packets from the bus (HSD2), reacts to the received packet, and interacts with its function(s) via the state of the corresponding endpoint (in the EP_array). Then the device may respond with a packet on the upstream bus (HSU1). The host controller state machines can receive a packet from the bus (HSU2) and provide a result of the transaction back to the host controller (HC_resp). The details of what packets are sent on the bus is determined by the transfer type for the endpoint and what bus activity the state machines observe.

The state machines are presented in a hierarchical form. Figure 8-19 shows the top level state machines for the host controller. The non-split transactions are presented in the remainder of this chapter. The split transaction state machines (HC_Do_start and HC_Do_complete) are described and shown in Chapter 11.

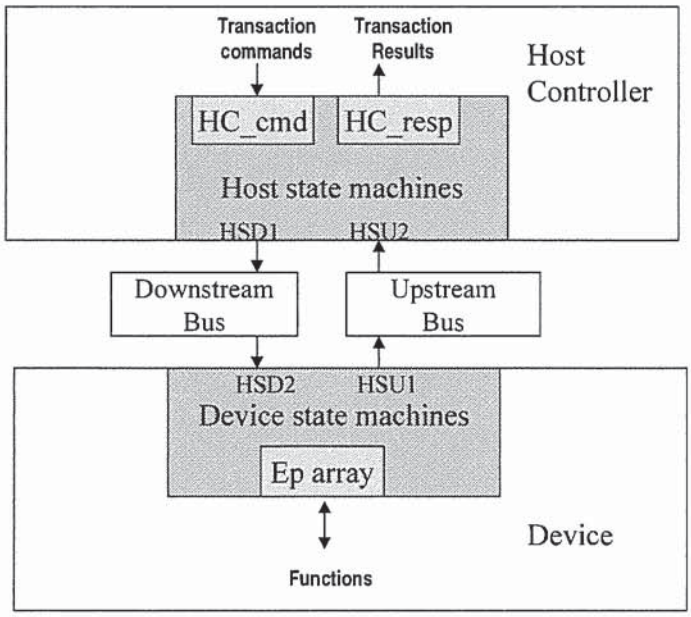


Figure 8-18. State Machine Context Overview

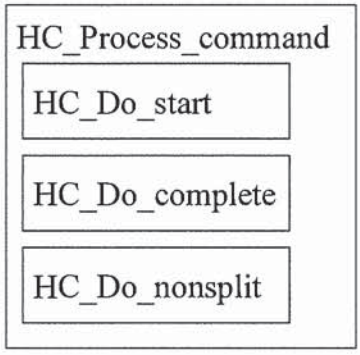


Figure 8-19. Host Controller Top Level Transaction State Machine Hierarchy Overview

The host controller state machines are located in the host controller. The host controller causes packets to be issued downstream (labeled as HSD1) and it receives upstream packets (labeled as HSU2).

The device state machines are located in the device. The device causes packets to be issued upstream (labeled as HSU1) and it receives downstream packets (labeled as HSD2).

The host controller has commands that tell it what transaction to issue next for an endpoint. The host controller tracks transactions for several endpoints. The host controller state machines sequence to determine what the host controller needs to do next for the current endpoint. The device has a state for each of its endpoints. The device state machines sequence to determine what reaction the device has to a transaction.

The appendix includes some declarations that were used in constructing the state machines and may be useful in understanding additional details of the state machines. There are several pseudo-code procedures and functions for conditions and actions. Simple descriptions of them are also included in the appendix.

Figure 8-20 shows an overview of the overall state machine hierarchy for the host controller for the non-split transaction types. Figure 8-21 shows the hierarchy of the device state machines. The state machines

common to endpoint types are presented first. The lowest level endpoint type specific state machines are presented in each following endpoint type section.

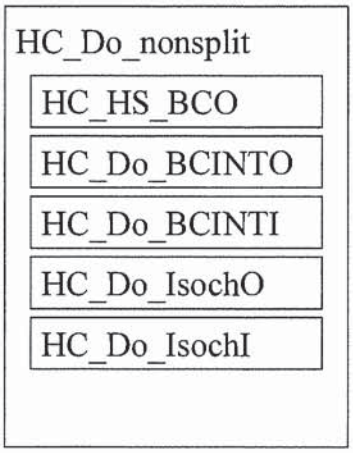


Figure 8-20. Host Controller Non-split Transaction State Machine Hierarchy Overview

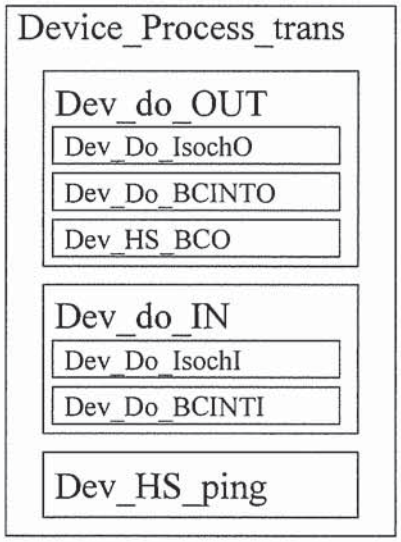


Figure 8-21. Device Transaction State Machine Hierarchy Overview

Global Actions | Concurrent Statements | Architecture Declarations | Signals Status | State Register Statements

Package List

ieee std_logic_1164
ieee numeric_std
usb2statemachines behav_package

```
SIGNAL  SCOPE DEFAULT
hsul    OUT   (BULK,NAK,0,0,ok,in_dir,TRUE,ALLDATA,FALSE,FA
device  INT   '0'
token   INT   '0'
```

Process Declarations

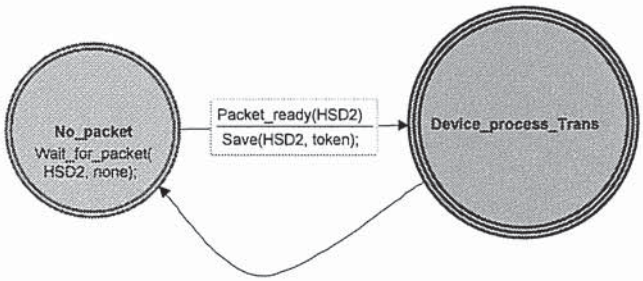


Figure 8-22. Device Top Level State Machine

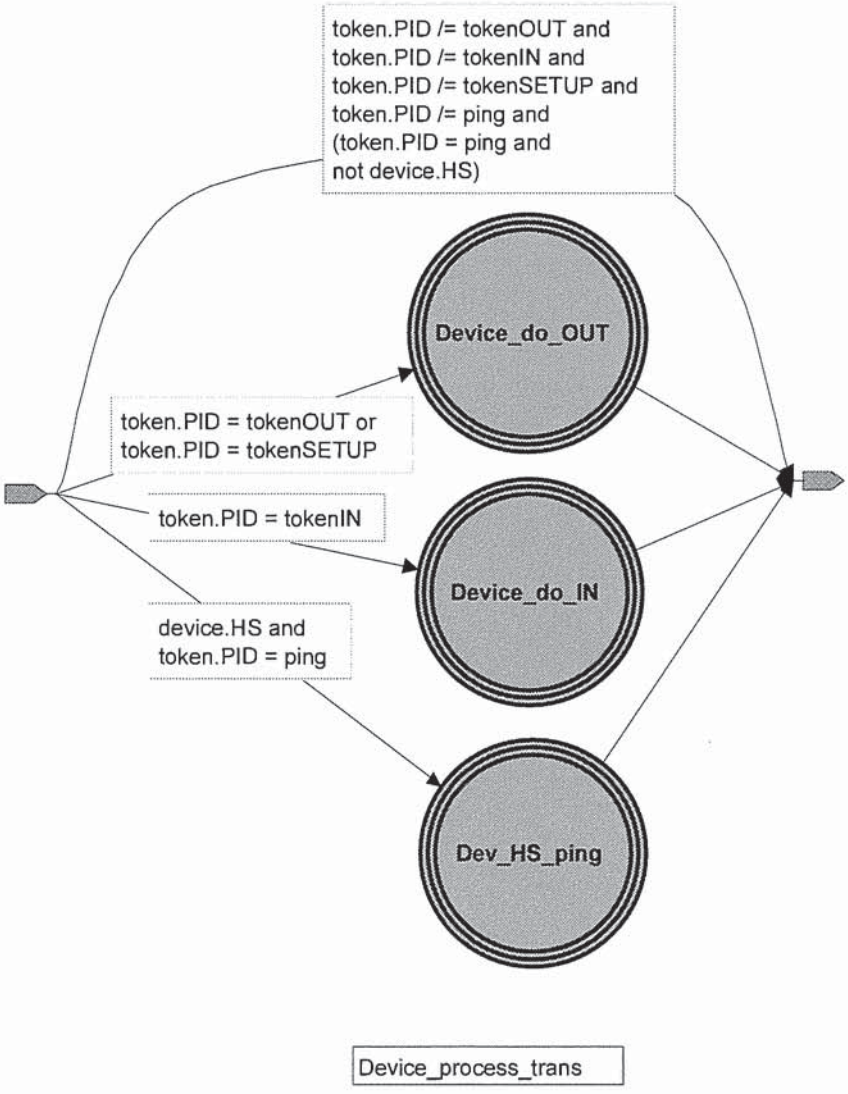


Figure 8-23. Device_process_Trans State Machine

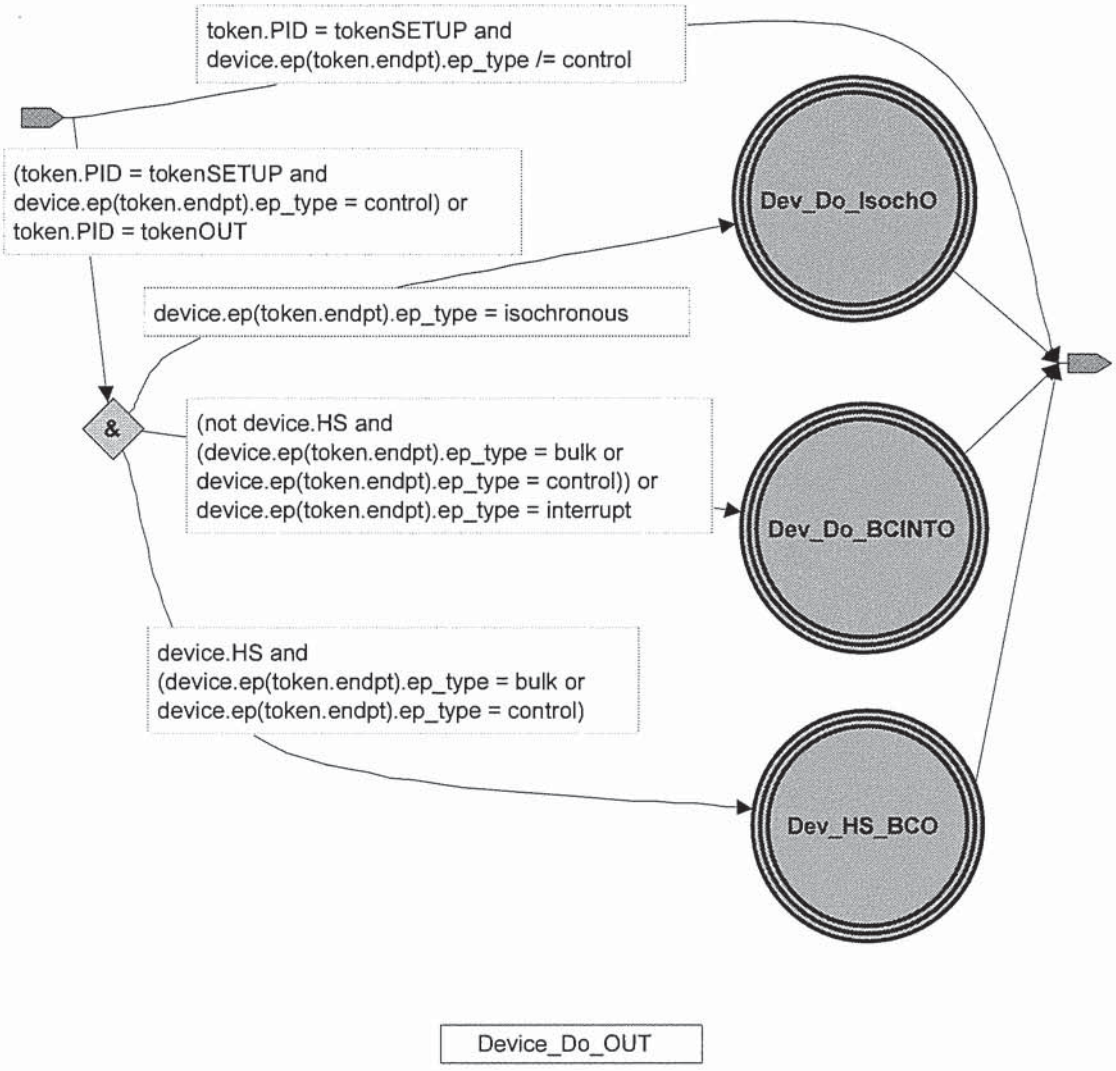


Figure 8-24. Dev_do_OUT State Machine

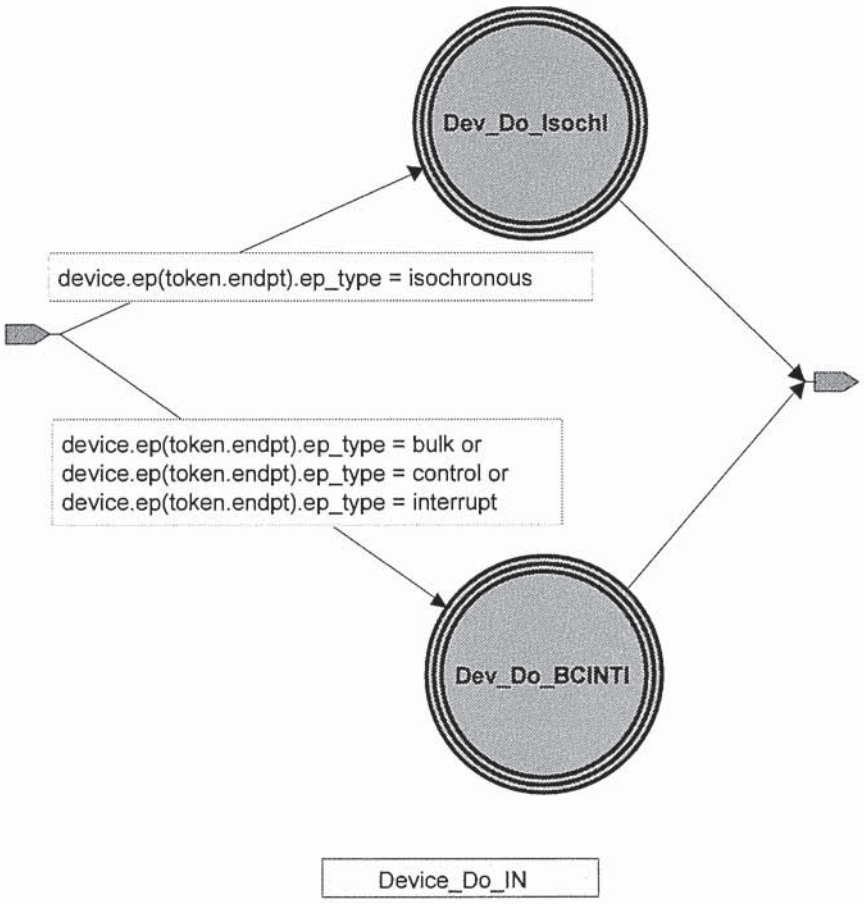


Figure 8-25. Dev_do_IN State Machine

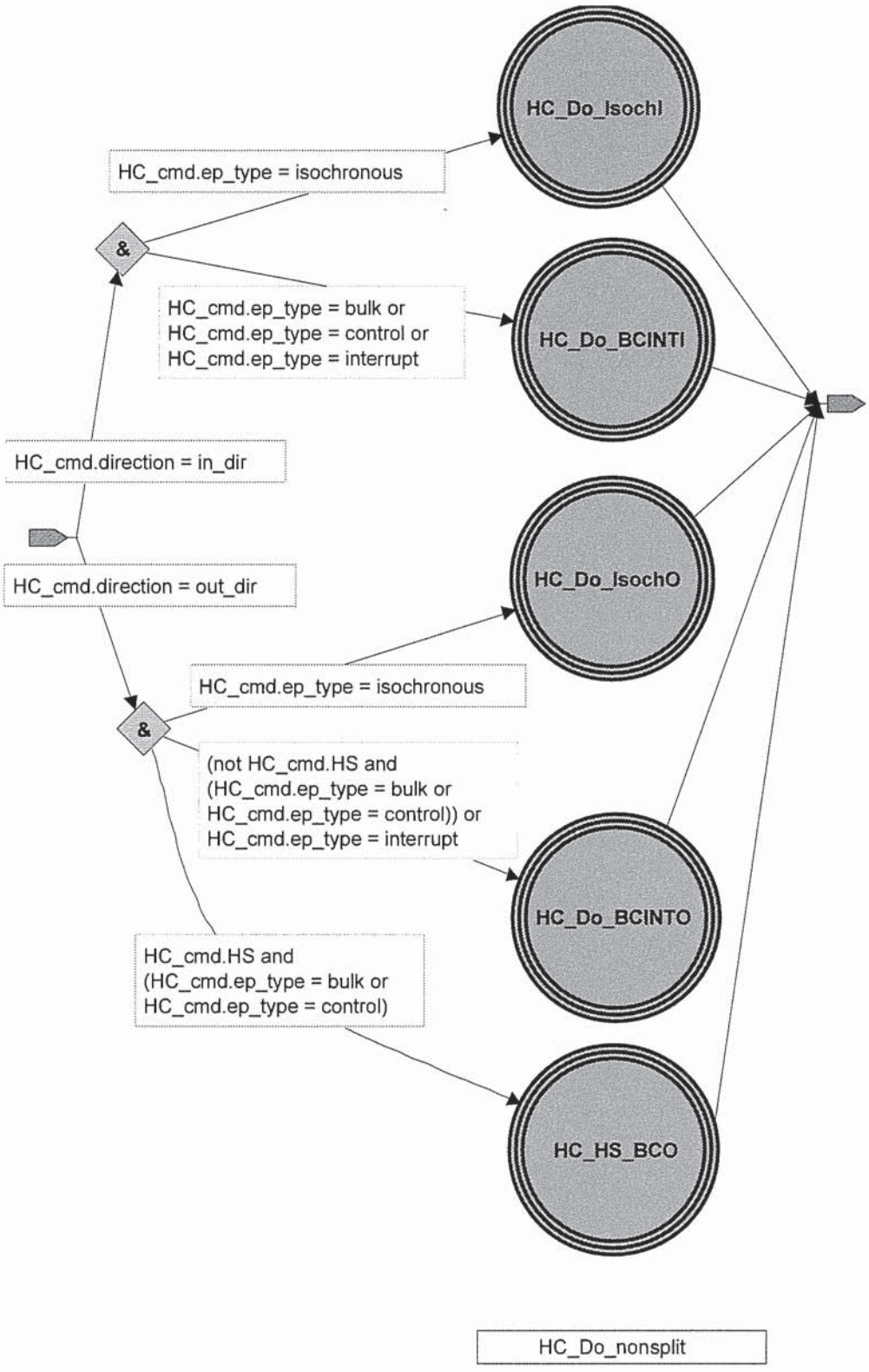


Figure 8-26. HC_Do_nonsplit State Machine

8.5.1 NAK Limiting via Ping Flow Control

Full-/low-speed devices can have bulk/control endpoints that take time to process their data and, therefore, respond to OUT transactions with a NAK handshake. This handshake response indicates that the endpoint did not accept the data because it did not have space for the data. The host controller is expected to retry the transaction at some future time when the endpoint has space available. Unfortunately, by the time the endpoint NAKs, most of the full-/low-speed bus time for the transaction had been used. This means that the full-/low-speed bus has poor utilization when there is a high frequency of NAK'd OUT transactions.

High-speed devices must support an improved NAK mechanism for Bulk OUT and Control endpoints and transactions. Control endpoints must support this protocol for an OUT transaction in the data and status stages. The control Setup stage must not support the PING protocol.

This mechanism allows the device to tell the host controller whether it has sufficient endpoint space for the next OUT transaction. If the device endpoint does not have space, the host controller can choose to delay a transaction attempt for this endpoint and instead try some other transaction. This can lead to improved bus utilization. The mechanism avoids using bus time to send data until the host controller knows that the endpoint has space for the data.

The host controller queries the high-speed device endpoint with a PING special token. The PING special token packet is a normal token packet as shown in Figure 8-5. The endpoint either responds to the PING with a NAK or an ACK handshake.

A NAK handshake indicates that the endpoint does not have space for a *wMaxPacketSize* data payload. The host controller will retry the PING at some future time to query the endpoint again. A device can respond to a PING with a NAK for long periods of time. A NAK response is not a reason for the host controller to retire a transfer request. If a device responds with a NAK in a (micro)frame, the host controller may choose to issue the next transaction in the next *bInterval* specified for the endpoint. However, the device must be prepared to receive PINGs as sequential transactions, e.g., one immediately after the other.

An ACK handshake indicates the endpoint has space for a *wMaxPacketSize* data payload. The host controller must generate an OUT transaction with a DATA phase as the next transaction to the endpoint. The host controller may generate other transactions to other devices or endpoints before the OUT/DATA transaction for this endpoint.

If the endpoint responds to the OUT/DATA transaction with an ACK handshake, this means the endpoint accepted the data successfully and has room for another *wMaxPacketSize* data payload. The host controller continues with OUT/DATA transactions (which are not required to be the next transactions on the bus) as long as it has transactions to generate.

If the endpoint instead responds to the OUT/DATA transaction with a NYET handshake, this means that the endpoint accepted the data but does not have room for another *wMaxPacketSize* data payload. The host controller must return to using a PING token until the endpoint indicates it has space.

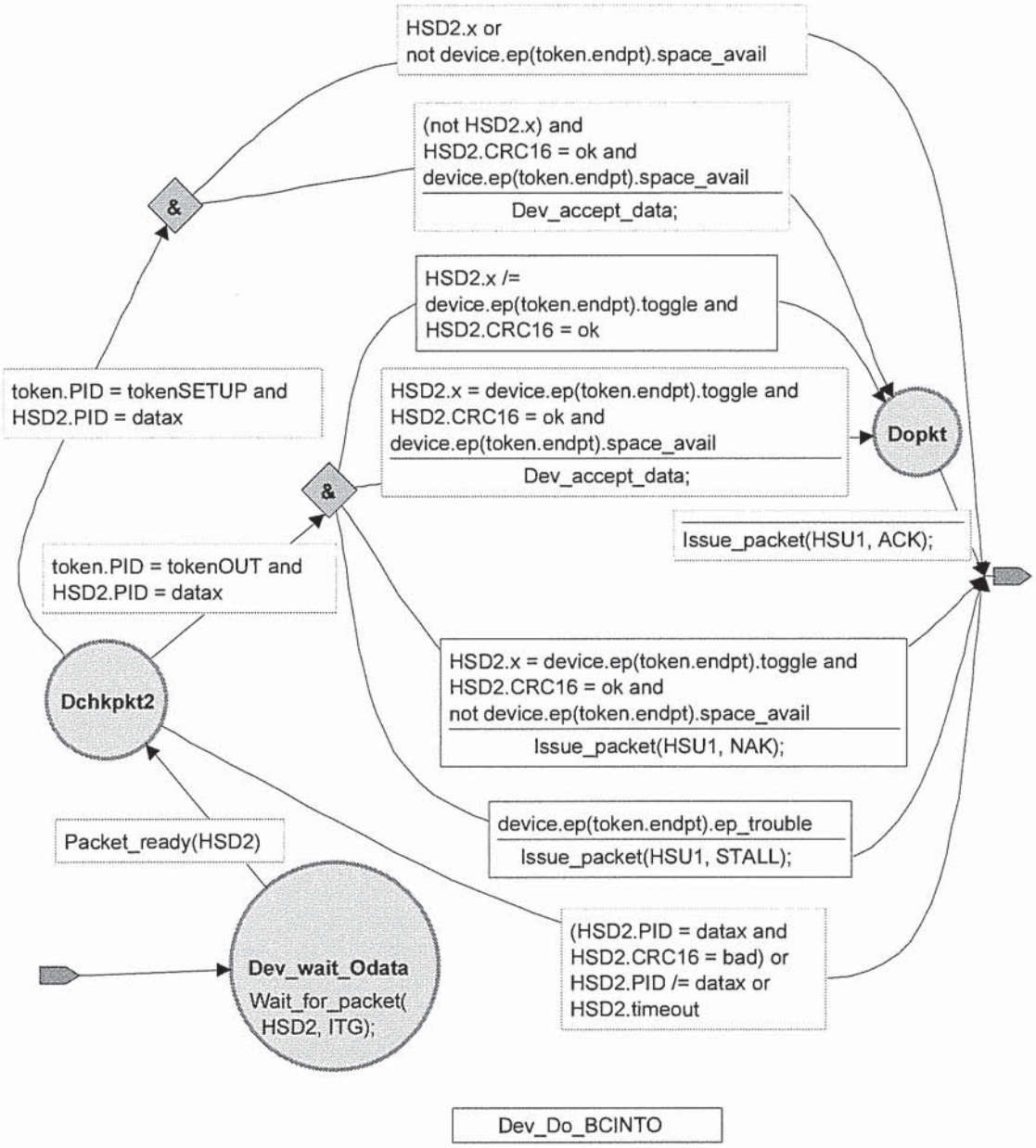


Figure 8-27. Host High-speed Bulk OUT/Control Ping State Machine

8.5.1.1 NAK Responses to OUT/DATA During PING Protocol

The endpoint may also respond to the OUT/DATA transaction with a NAK handshake. This means that the endpoint did not accept the data and does not have space for a *wMaxPacketSize* data payload at this time. The host controller must return to using a PING token until the endpoint indicates it has space.

A NAK response is expected to be an unusual occurrence. A high-speed bulk/control endpoint must specify its maximum NAK rate in its endpoint descriptor. The endpoint is allowed to NAK at most one time each *bInterval* period. A NAK suggests that the endpoint responded to a previous OUT or PING with an inappropriate handshake, or that the endpoint transitioned into a state where it (temporarily) could not

accept data. An endpoint can use a *bInterval* of zero to indicate that it never NAKs. An endpoint must always be able to accept a PING from the host, even if it never NAKs.

If a timeout occurs after the data phase, the host must return to using a PING token. Note that a transition back to the PING state does not affect the data toggle state of the transaction data phase.

Figure 8-27 shows the host controller state machine for the interactions and transitions between PING and OUT/DATA tokens and the allowed ACK, NAK, and NYET handshakes for the PING mechanism.

Figure 8-29 shows the device endpoint state machine for PING based on the buffer space the endpoint has available.

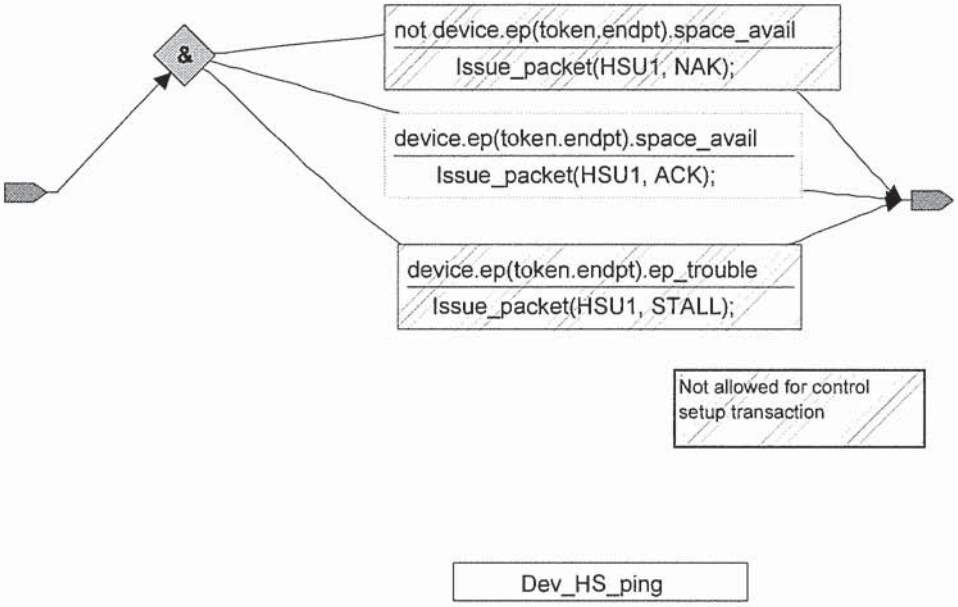


Figure 8-28. Dev_HS_ping State Machine

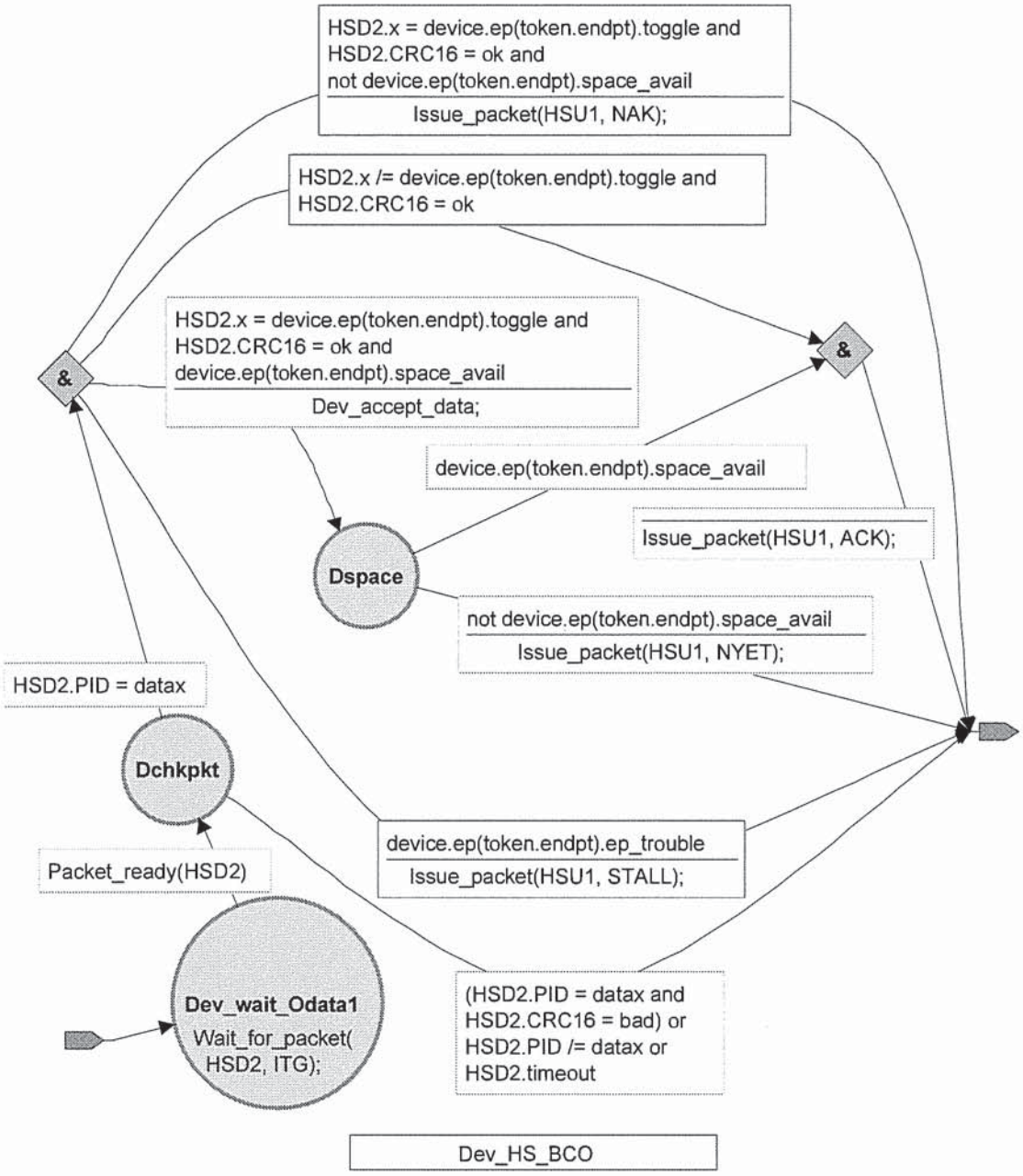


Figure 8-29. Device High-speed Bulk OUT /Control State Machine

Full-/low-speed devices/endpoints must not support the PING protocol. Host controllers must not support the PING protocol for full-/low-speed devices.

Note: The PING protocol is also not included as part of the split-transaction protocol definition. Some split-transactions have equivalent flow control without using PING. Other split-transactions will not benefit from PING as defined. In any case, split-transactions that can return a NAK handshake have small data payloads which should have minor high-speed bus impact. Hubs must support PING on their control endpoint, but PING is not defined for the split-transactions that are used to communicate with full-/low-speed devices supported by a hub.

8.5.2 Bulk Transactions

Bulk transaction types are characterized by the ability to guarantee error-free delivery of data between the host and a function by means of error detection and retry. Bulk transactions use a three-phase transaction consisting of token, data, and handshake packets as shown in Figure 8-30. Under certain flow control and halt conditions, the data phase may be replaced with a handshake resulting in a two-phase transaction in which no data is transmitted. The PING and NYET packets must only be used with devices operating at high-speed.

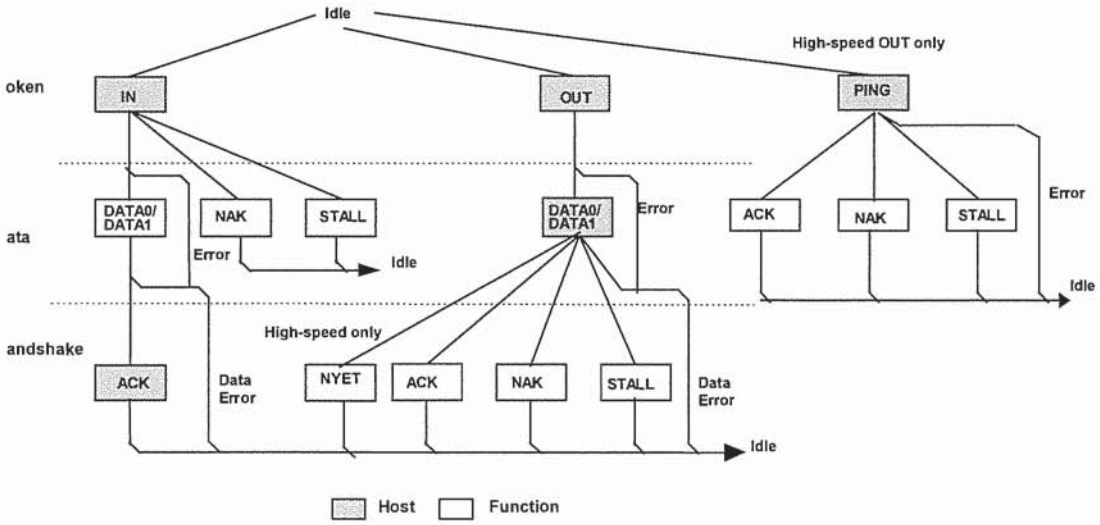


Figure 8-30. Bulk Transaction Format

When the host is ready to receive bulk data, it issues an IN token. The function endpoint responds by returning either a data packet or, should it be unable to return data, a NAK or STALL handshake. NAK indicates that the function is temporarily unable to return data, while STALL indicates that the endpoint is permanently halted and requires USB System Software intervention. If the host receives a valid data packet, it responds with an ACK handshake. If the host detects an error while receiving data, it returns no handshake packet to the function.

When the host is ready to transmit bulk data, it first issues an OUT token packet followed by a data packet (or PING special token packet, see Section 8.5.1). If the data is received without error by the function, it will return one of three (or four including NYET, for a device operating at high-speed) handshakes:

- ACK indicates that the data packet was received without errors and informs the host that it may send the next packet in the sequence.
- NAK indicates that the data was received without error but that the host should resend the data because the function was in a temporary condition preventing it from accepting the data (e.g., buffer full).
- If the endpoint was halted, STALL is returned to indicate that the host should not retry the transmission because there is an error condition on the function.

If the data packet was received with a CRC or bit stuff error, no handshake is returned.

Figure 8-31 and Figure 8-32 show the host and device state machines respectively for bulk, control, and interrupt OUT full/low-speed transactions. Figure 8-27, Figure 8-28, and Figure 8-29 show the state machines for high-speed transactions. Figure 8-33 and Figure 8-34 show the host and device state machines respectively for bulk, control, and interrupt IN transactions.

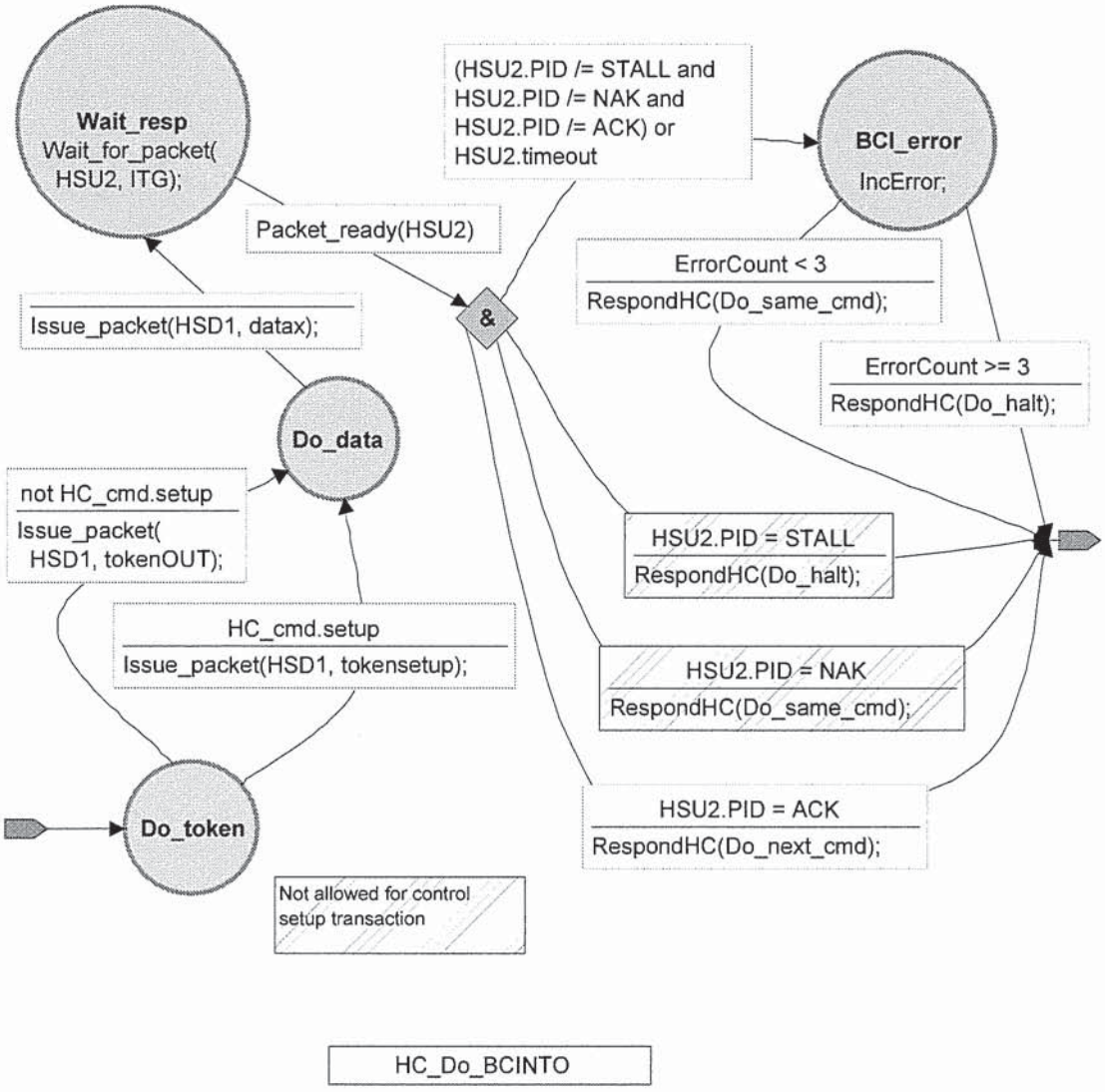


Figure 8-31. Bulk/Control/Interrupt OUT Transaction Host State Machine

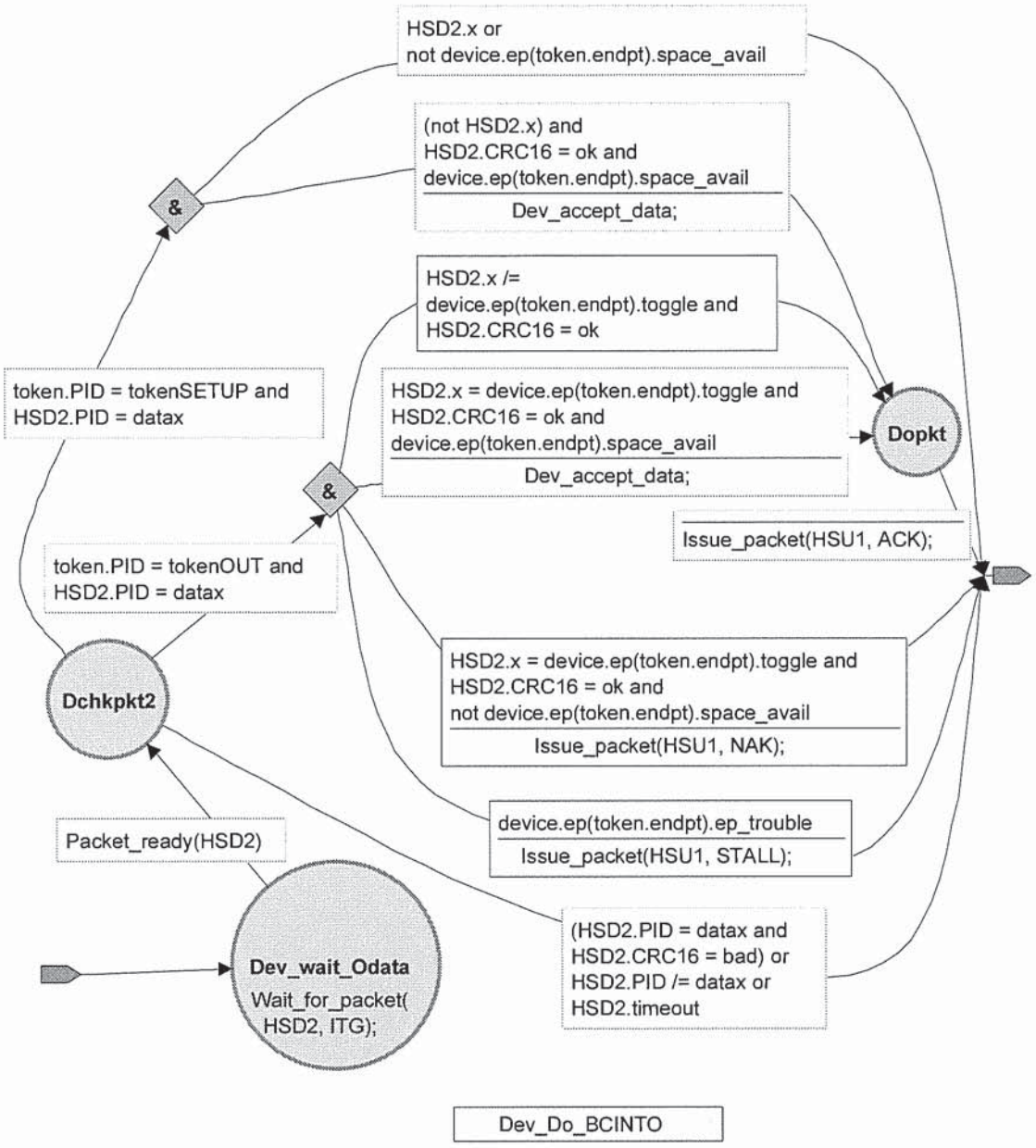


Figure 8-32. Bulk/Control/Interrupt OUT Transaction Device State Machine

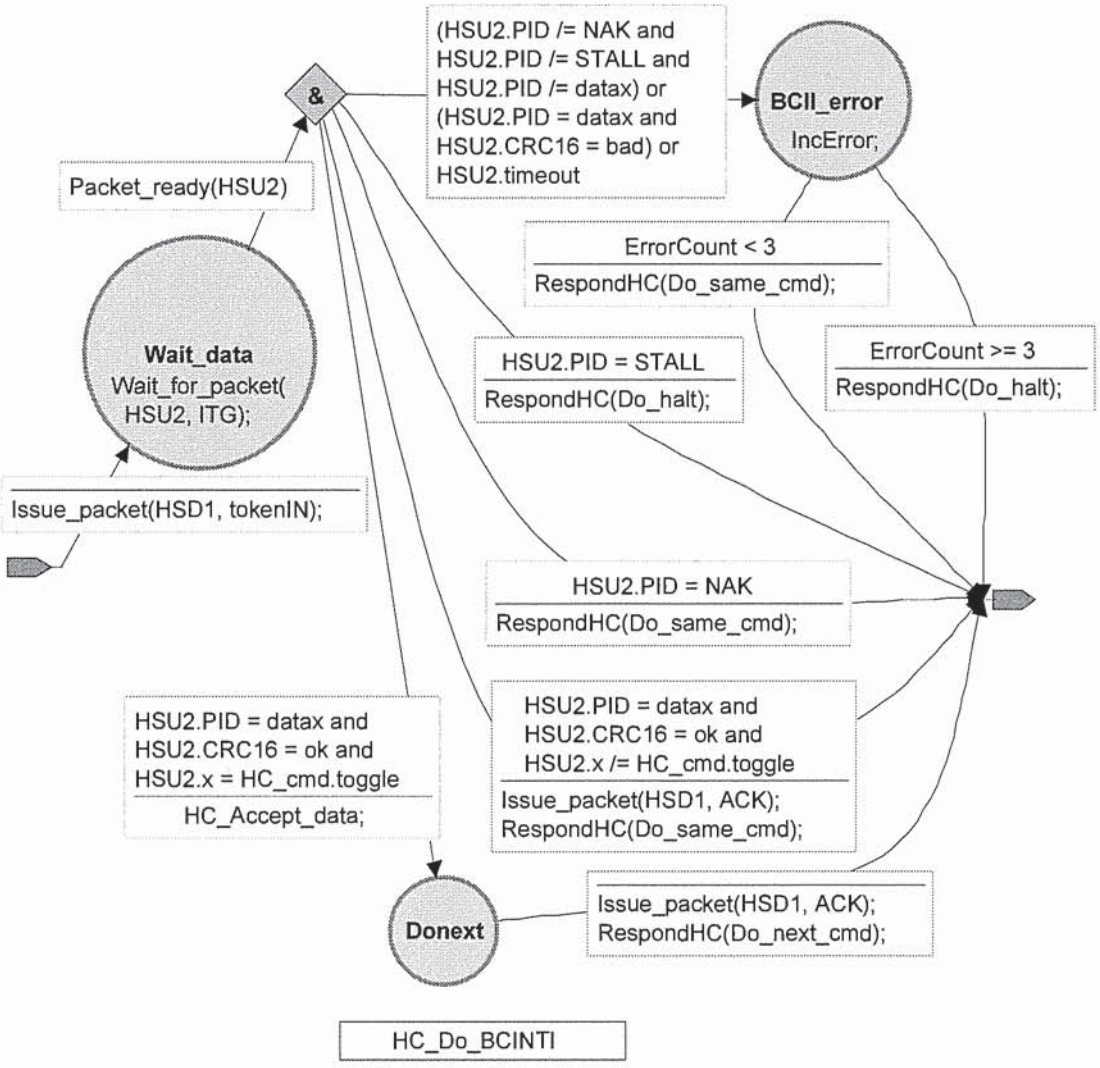


Figure 8-33. Bulk/Control/Interrupt IN Transaction Host State Machine

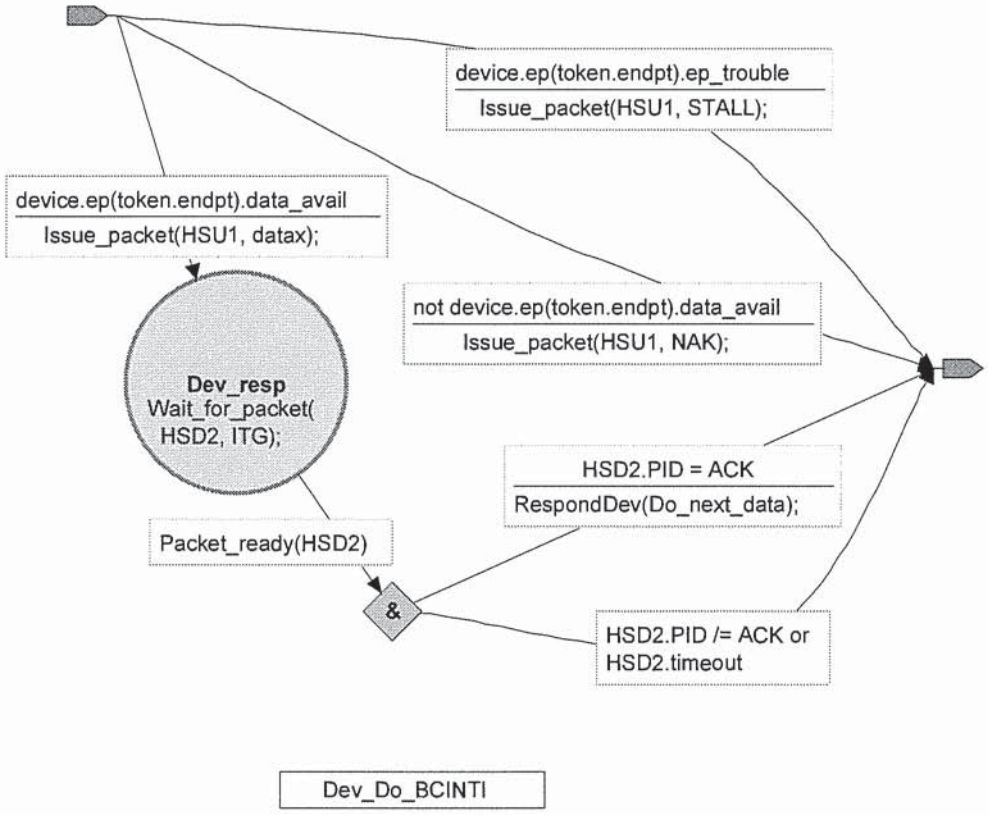


Figure 8-34. Bulk/Control/Interrupt IN Transaction Device State Machine

Figure 8-35 shows the sequence bit and data PID usage for bulk reads and writes. Data packet synchronization is achieved via use of the data sequence toggle bits and the DATA0/DATA1 PIDs. A bulk endpoint's toggle sequence is initialized to DATA0 when the endpoint experiences any configuration event (configuration events are explained in Sections 9.1.1.5 and 9.4.5). Data toggle on an endpoint is NOT initialized as the direct result of a short packet transfer or the retirement of an IRP.

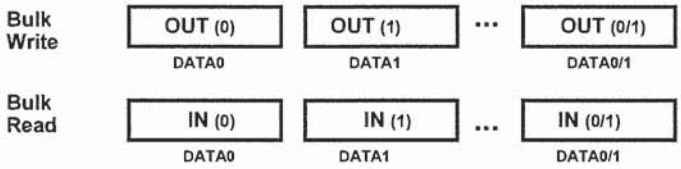


Figure 8-35. Bulk Reads and Writes

The host always initializes the first transaction of a bus transfer to the DATA0 PID with a configuration event. The second transaction uses a DATA1 PID, and successive data transfers alternate for the remainder of the bulk transfer. The data packet transmitter toggles upon receipt of ACK, and the receiver toggles upon receipt and acceptance of a valid data packet (refer to Section 8.6).

8.5.3 Control Transfers

Control transfers minimally have two transaction stages: Setup and Status. A control transfer may optionally contain a Data stage between the Setup and Status stages. During the Setup stage, a SETUP transaction is used to transmit information to the control endpoint of a function. SETUP transactions are similar in format to an OUT but use a SETUP rather than an OUT PID. Figure 8-36 shows the SETUP transaction format. A SETUP always uses a DATA0 PID for the data field of the SETUP transaction. The

function receiving a SETUP must accept the SETUP data and respond with ACK; if the data is corrupted, discard the data and return no handshake.

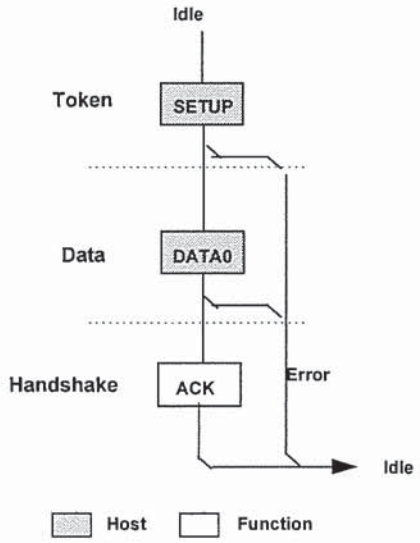


Figure 8-36. Control SETUP Transaction

The Data stage, if present, of a control transfer consists of one or more IN or OUT transactions and follows the same protocol rules as bulk transfers. All the transactions in the Data stage must be in the same direction (i.e., all INs or all OUTs). The amount of data to be sent during the data stage and its direction are specified during the Setup stage. If the amount of data exceeds the prenegotiated data packet size, the data is sent in multiple transactions (INs or OUTs) that carry the maximum packet size. Any remaining data is sent as a residual in the last transaction.

The Status stage of a control transfer is the last transaction in the sequence. The status stage transactions follow the same protocol sequence as bulk transactions. Status stage for devices operating at high-speed also includes the PING protocol. A Status stage is delineated by a change in direction of data flow from the previous stage and always uses a DATA1 PID. If, for example, the Data stage consists of OUTs, the status is a single IN transaction. If the control sequence has no Data stage, then it consists of a Setup stage followed by a Status stage consisting of an IN transaction.

Figure 8-37 shows the transaction order, the data sequence bit value, and the data PID types for control read and write sequences. The sequence bits are displayed in parentheses.

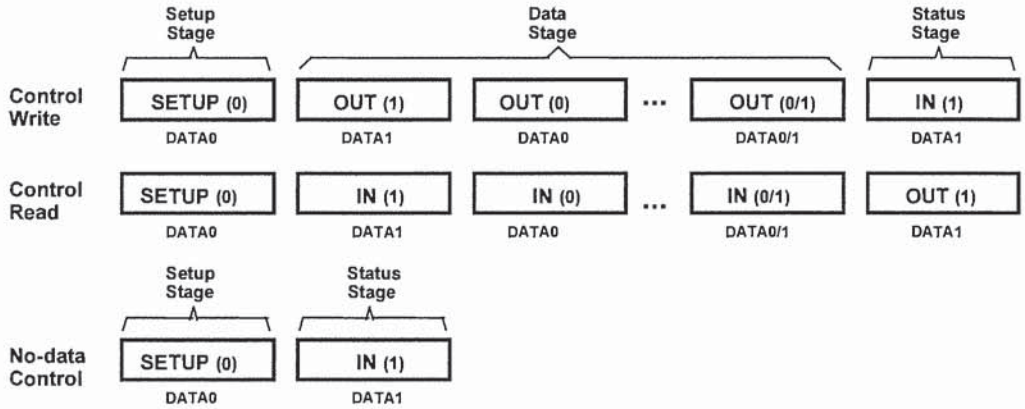


Figure 8-37. Control Read and Write Sequences

When a STALL handshake is sent by a control endpoint in either the Data or Status stages of a control transfer, a STALL handshake must be returned on all succeeding accesses to that endpoint until a SETUP PID is received. The endpoint is not required to return a STALL handshake after it receives a subsequent SETUP PID. For the default endpoint, if an ACK handshake is returned for the SETUP transaction, the host expects that the endpoint has automatically recovered from the condition that caused the STALL and the endpoint must operate normally.

8.5.3.1 Reporting Status Results

The Status stage reports to the host the outcome of the previous Setup and Data stages of the transfer. Three possible results may be returned:

- The command sequence completed successfully.
- The command sequence failed to complete.
- The function is still busy completing the command.

Status reporting is always in the function-to-host direction. Table 8-7 summarizes the type of responses required for each. Control write transfers return status information in the data phase of the Status stage transaction. Control read transfers return status information in the handshake phase of a Status stage transaction, after the host has issued a zero-length data packet during the previous data phase.

Table 8-7. Status Stage Responses

Status Response	Control Write Transfer (sent during data phase)	Control Read Transfer (sent during handshake phase)
Function completes	Zero-length data packet	ACK handshake
Function has an error	STALL handshake	STALL handshake
Function is busy	NAK handshake	NAK handshake

For control reads, the host must send either an OUT token or PING special token (for a device operating at high-speed) to the control pipe to initiate the Status stage. The host may only send a zero-length data packet in this phase but the function may accept any length packet as a valid status inquiry. The pipe's handshake response to this data packet indicates the current status. NAK indicates that the function is still processing the command and that the host should continue the Status stage. ACK indicates that the function has completed the command and is ready to accept a new command. STALL indicates that the function has an error that prevents it from completing the command.

For control writes, the host sends an IN token to the control pipe to initiate the Status stage. The function responds with either a handshake or a zero-length data packet to indicate its current status. NAK indicates that the function is still processing the command and that the host should continue the Status stage; return of a zero-length packet indicates normal completion of the command; and STALL indicates that the function cannot complete the command. The function expects the host to respond to the data packet in the Status stage with ACK. If the function does not receive ACK, it remains in the Status stage of the command and will continue to return the zero-length data packet for as long as the host continues to send IN tokens.

If during a Data stage a command pipe is sent more data or is requested to return more data than was indicated in the Setup stage (see Section 8.5.3.2), it should return STALL. If a control pipe returns STALL during the Data stage, there will be no Status stage for that control transfer.

8.5.3.2 Variable-length Data Stage

A control pipe may have a variable-length data phase in which the host requests more data than is contained in the specified data structure. When all of the data structure is returned to the host, the function should indicate that the Data stage is ended by returning a packet that is shorter than the *MaxPacketSize* for the pipe. If the data structure is an exact multiple of *wMaxPacketSize* for the pipe, the function will return a zero-length packet to indicate the end of the Data stage.

8.5.3.3 Error Handling on the Last Data Transaction

If the ACK handshake on an IN transaction is corrupted, the function and the host will temporarily disagree on whether the transaction was successful. If the transaction is followed by another IN, the toggle retry mechanism will detect the mismatch and recover from the error. If the ACK was on the last IN of a Data stage, the toggle retry mechanism cannot be used and an alternative scheme must be used.

The host that successfully received the data of the last IN will send ACK. Later, the host will issue an OUT token to start the Status stage of the transfer. If the function did not receive the ACK that ended the Data stage, the function will interpret the start of the Status stage as verification that the host successfully received the data. Control writes do not have this ambiguity. If an ACK handshake on an OUT gets corrupted, the host does not advance to the Status stage and retries the last data instead. A detailed analysis of retry policy is presented in Section 8.6.4.

8.5.3.4 STALL Handshakes Returned by Control Pipes

Control pipes have the unique ability to return a STALL handshake due to function problems in control transfers. If the device is unable to complete a command, it returns a STALL in the Data and/or Status stages of the control transfer. Unlike the case of a functional stall, protocol stall does not indicate an error with the device. The protocol STALL condition lasts until the receipt of the next SETUP transaction, and the function will return STALL in response to any IN or OUT transaction on the pipe until the SETUP transaction is received. In general, protocol stall indicates that the request or its parameters are not understood by the device and thus provides a mechanism for extending USB requests.

A control pipe may also support functional stall as well, but this is not recommended. This is a degenerative case, because a functional stall on a control pipe indicates that it has lost the ability to communicate with the host. If the control pipe does support functional stall, then it must possess a *Halt* feature, which can be set or cleared by the host. Chapter 9 details how to treat the special case of a *Halt* feature on a control pipe. A well-designed device will associate all of its functions and *Halt* features with non-control endpoints. The control pipes should be reserved for servicing USB requests.

8.5.4 Interrupt Transactions

Interrupt transactions may consist of IN or OUT transfers. Upon receipt of an IN token, a function may return data, NAK, or STALL. If the endpoint has no new interrupt information to return (i.e., no interrupt is pending), the function returns a NAK handshake during the data phase. If the *Halt* feature is set for the interrupt endpoint, the function will return a STALL handshake. If an interrupt is pending, the function returns the interrupt information as a data packet. The host, in response to receipt of the data packet, issues either an ACK handshake if data was received error-free or returns no handshake if the data packet was received corrupted. Figure 8-38 shows the interrupt transaction format.

Section 5.9.1 contains additional information about high-speed, high-bandwidth interrupt endpoints. Such endpoints use multiple transactions in a microframe as defined in that section. Each transaction for a high-bandwidth endpoint follows the transaction format shown in Figure 8-38.

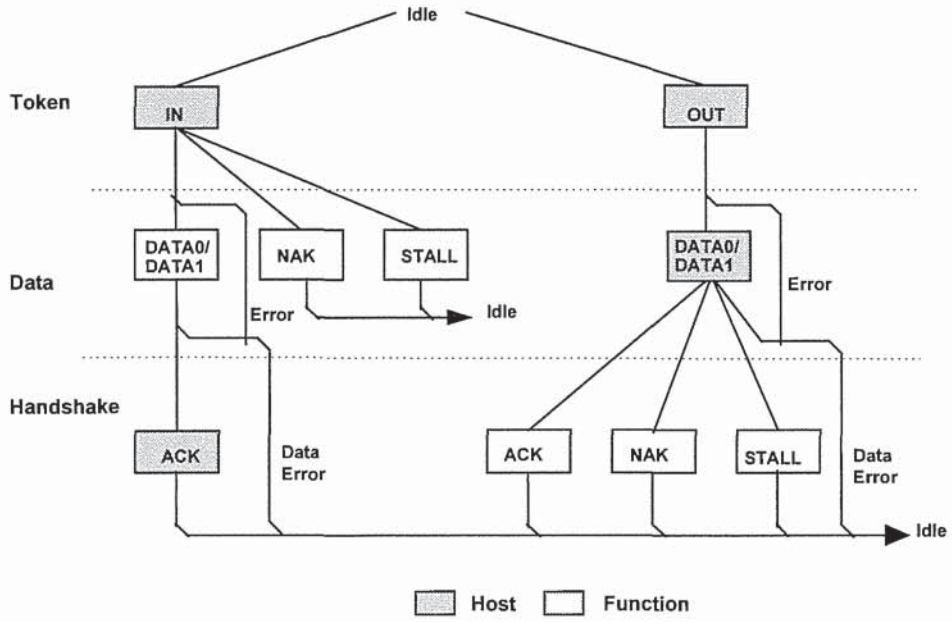


Figure 8-38. Interrupt Transaction Format

When an endpoint is using the interrupt transfer mechanism for actual interrupt data, the data toggle protocol must be followed. This allows the function to know that the data has been received by the host and the event condition may be cleared. This "guaranteed" delivery of events allows the function to only send the interrupt information until it has been received by the host rather than having to send the interrupt data every time the function is polled and until the USB System Software clears the interrupt condition. When used in the toggle mode, an interrupt endpoint is initialized to the DATA0 PID by any configuration event on the endpoint and behaves the same as the bulk transactions shown in Figure 8-35.

8.5.5 Isochronous Transactions

Isochronous transactions have a token and data phase, but no handshake phase, as shown in Figure 8-39. The host issues either an IN or an OUT token followed by the data phase in which the endpoint (for INs) or the host (for OUTs) transmits data. Isochronous transactions do not support a handshake phase or retry capability.

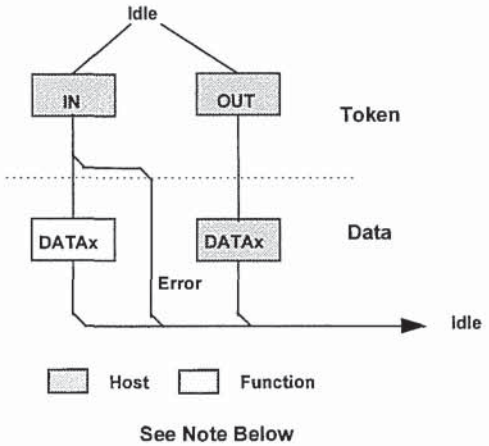


Figure 8-39. Isochronous Transaction Format

Note: A full-speed device or Host Controller should be able to accept either DATA0 or DATA1 PIDs in data packets. A full-speed device or Host Controller should only send DATA0 PIDs in data packets. A high-speed Host Controller must be able to accept and send DATA0, DATA1, DATA2, or MDATA PIDs in data packets. A high-speed device with at most 1 transaction per microframe must only send DATA0 PIDs in data packets. A high-speed device with high-bandwith endpoints (e.g., one that has more than 1 transaction per microframe) must be able to accept and/or send DATA0, DATA1, DATA2, or MDATA PIDs in data packets.

Full-speed isochronous transactions do not support toggle sequencing. High-speed isochronous transactions with a single transaction per microframe do not support toggle sequencing. High bandwidth, high-speed isochronous transactions support data PID sequencing (see Section 5.9.1 for more details).

Figure 8-40 and Figure 8-41 show the host and device state machines respectively for isochronous OUT transactions. Figure 8-42 and Figure 8-43 show the host and device state machines respectively for isochronous IN transactions.

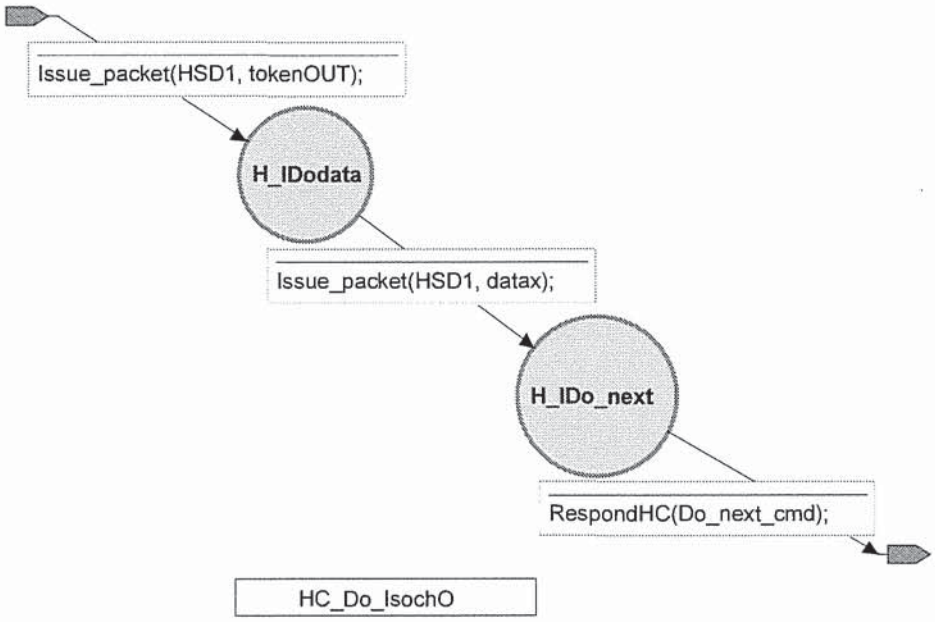


Figure 8-40. Isochronous OUT Transaction Host State Machine

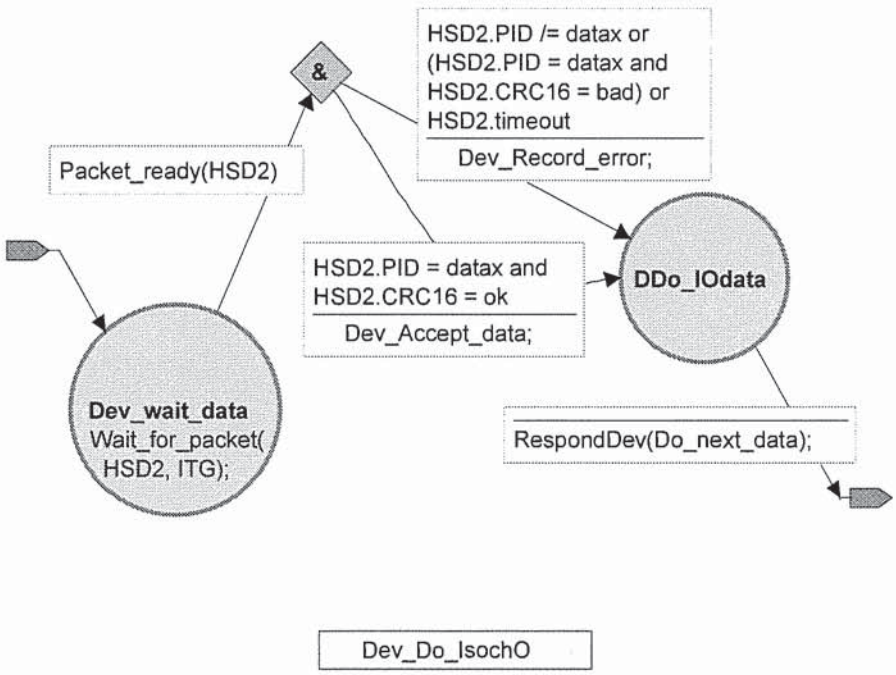


Figure 8-41. Isochronous OUT Transaction Device State Machine

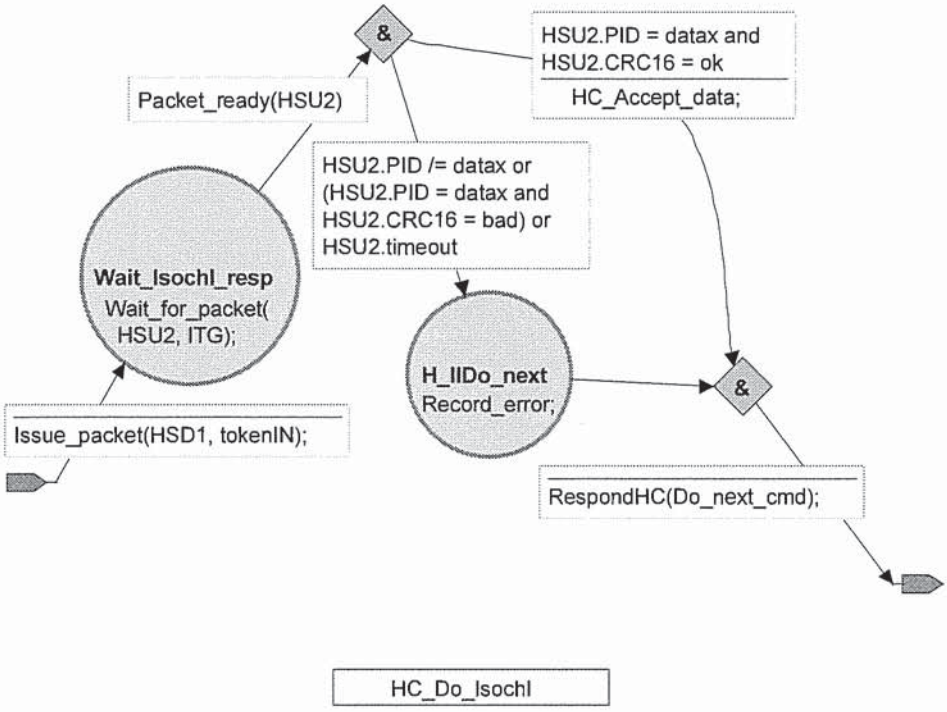


Figure 8-42. Isochronous IN Transaction Host State Machine

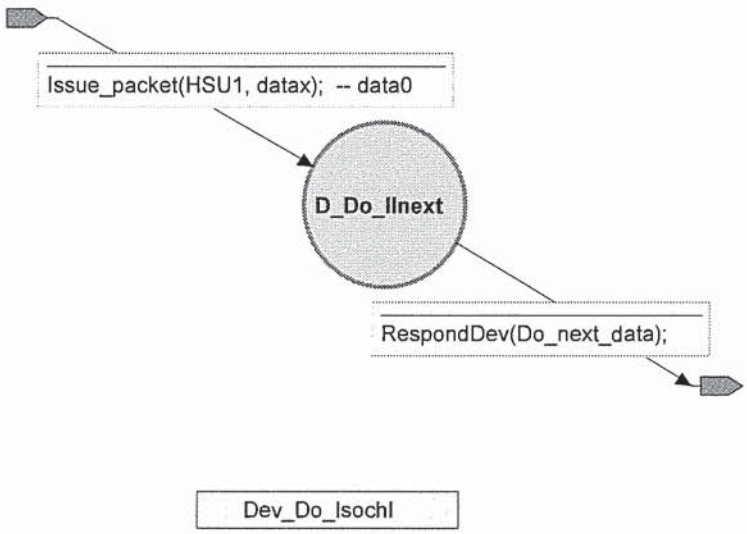


Figure 8-43. Isochronous IN Transaction Device State Machine

8.6 Data Toggle Synchronization and Retry

The USB provides a mechanism to guarantee data sequence synchronization between data transmitter and receiver across multiple transactions. This mechanism provides a means of guaranteeing that the handshake phase of a transaction was interpreted correctly by both the transmitter and receiver. Synchronization is achieved via use of the DATA0 and DATA1 PIDs and separate data toggle sequence bits for the data transmitter and receiver. Receiver sequence bits toggle only when the receiver is able to accept data and receives an error-free data packet with the correct data PID. Transmitter sequence bits toggle only when the data transmitter receives a valid ACK handshake. The data transmitter and receiver must have their sequence bits synchronized at the start of a transaction. The synchronization mechanism used varies with the transaction type. Data toggle synchronization is not supported for isochronous transfers.

The state machines contained in this chapter and in Chapter 11 describe data toggle synchronization in a more compact form. Instead of explicitly identifying DATA0 and DATA1, it uses a value "DATAx" to represent either/both DATA0/DATA1 PIDs. In some cases where the specific data PID is important, another variable labeled "x" is used that has the value 0 for DATA0 and 1 for DATA1.

High-speed, high-bandwidth isochronous and interrupt endpoints support a similar but different data synchronization technique called data PID sequencing. That technique is used instead of data toggle synchronization. Section 5.9.1 defines data PID sequencing.

8.6.1 Initialization via SETUP Token

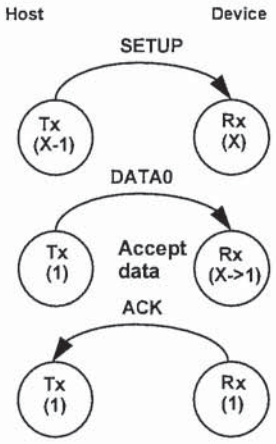


Figure 8-44. SETUP Initialization

Control transfers use the SETUP token for initializing host and function sequence bits. Figure 8-44 shows the host issuing a SETUP packet to a function followed by an OUT transaction. The numbers in the circles represent the transmitter and receiver sequence bits. The function must accept the data and return ACK. When the function accepts the transaction, it must set its sequence bit so that both the host's and function's sequence bits are equal to one at the end of the SETUP transaction.

8.6.2 Successful Data Transactions

Figure 8-45 shows the case where two successful transactions have occurred. For the data transmitter, this means that it toggles its sequence bit upon receipt of ACK. The receiver toggles its sequence bit only if it receives a valid data packet and the packet's data PID matches the current value of its sequence bit. The transmitter only toggles its sequence bit after it receives an ACK to a data packet.

During each transaction, the receiver compares the transmitter sequence bit (encoded in the data packet PID as either DATA0 or DATA1) with its receiver sequence bit. If data cannot be accepted, the receiver must issue NAK and the sequence bits of both the transmitter and receiver remain unchanged. If data can be accepted and the receiver's sequence bit matches the PID sequence bit, then data is accepted and the sequence bit is toggled. Two-phase transactions in which there is no data packet leave the transmitter and receiver sequence bits unchanged.

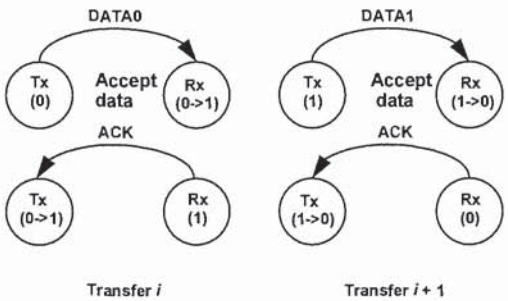


Figure 8-45. Consecutive Transactions

8.6.3 Data Corrupted or Not Accepted

If data cannot be accepted or the received data packet is corrupted, the receiver will issue a NAK or STALL handshake, or timeout, depending on the circumstances, and the receiver will not toggle its sequence bit.

Figure 8-46 shows the case where a transaction is NAKed and then retried. Any non-ACK handshake or timeout will generate similar retry behavior. The transmitter, having not received an ACK handshake, will not toggle its sequence bit. As a result, a failed data packet transaction leaves the transmitter's and receiver's sequence bits synchronized and untoggled. The transaction will then be retried and, if successful, will cause both transmitter and receiver sequence bits to toggle.

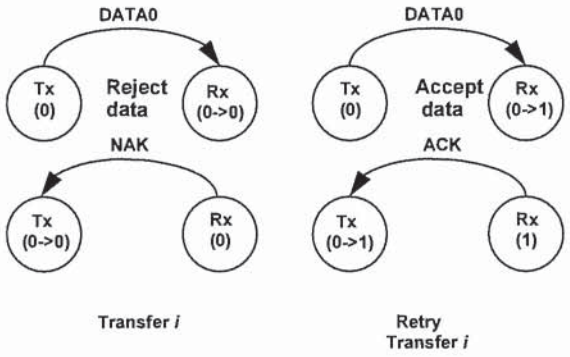


Figure 8-46. NAKed Transaction with Retry

8.6.4 Corrupted ACK Handshake

The transmitter is the last and only agent to know for sure whether a transaction has been successful, due to its receiving an ACK handshake. A lost or corrupted ACK handshake can lead to a temporary loss of synchronization between transmitter and receiver as shown in Figure 8-47. Here the transmitter issues a valid data packet, which is successfully acquired by the receiver; however, the ACK handshake is corrupted.

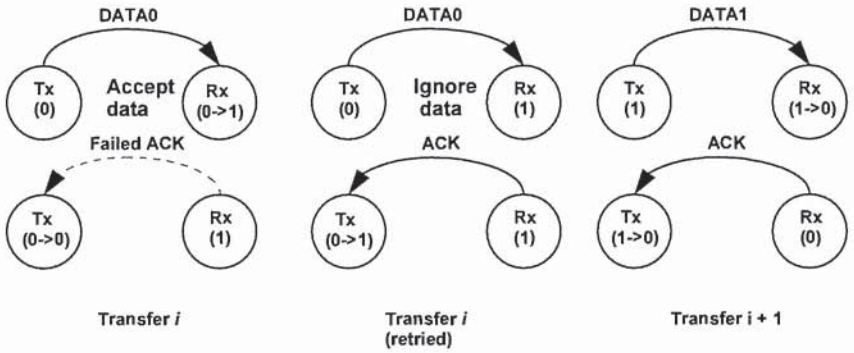


Figure 8-47. Corrupted ACK Handshake with Retry

At the end of transaction *i*, there is a temporary loss of coherency between transmitter and receiver, as evidenced by the mismatch between their respective sequence bits. The receiver has received good data, but the transmitter does not know whether it has successfully sent data. On the next transaction, the transmitter will resend the previous data using the previous DATA0 PID. The receiver's sequence bit and the data PID will not match, so the receiver knows that it has previously accepted this data. Consequently, it discards the incoming data packet and does not toggle its sequence bit. The receiver then issues ACK, which causes the transmitter to regard the retried transaction as successful. Receipt of ACK causes the transmitter to toggle its sequence bit. At the beginning of transaction *i*+1, the sequence bits have toggled and are again synchronized.

The data transmitter must guarantee that any retried data packet is identical (same length and content) as that sent in the original transaction. If the data transmitter is unable, because of problems such as a buffer underrun condition, to transmit the identical amount of data as was in the original data packet, it must abort

the transaction by generating a bit stuffing violation for full-/low-speed. An error for high-speed must be forced by taking the currently calculated CRC and complementing it before transmitting it. This causes a detectable error at the receiver and guarantees that a partial packet will not be interpreted as a good packet. The transmitter should not try to force an error at the receiver by sending a constant known bad CRC. A combination of a bad packet with a "bad" CRC may be interpreted by the receiver as a good packet.

8.6.5 Low-speed Transactions

The USB supports signaling at three speeds: high-speed signaling at 480 Mb/s, full-speed signaling at 12.0 Mb/s, and low-speed signaling at 1.5 Mb/s. Hubs isolate high-speed signaling from full-/low-speed signaling environments.

Within a full-/low-speed signaling environment, hubs disable downstream bus traffic to all ports to which low-speed devices are attached during full-speed downstream signaling. This is required both for EMI reasons and to prevent any possibility that a low-speed device might misinterpret downstream a full-speed packet as being addressed to it.

Figure 8-48 shows an IN low-speed transaction in which the host (or TT) issues a token and handshake and receives a data packet.

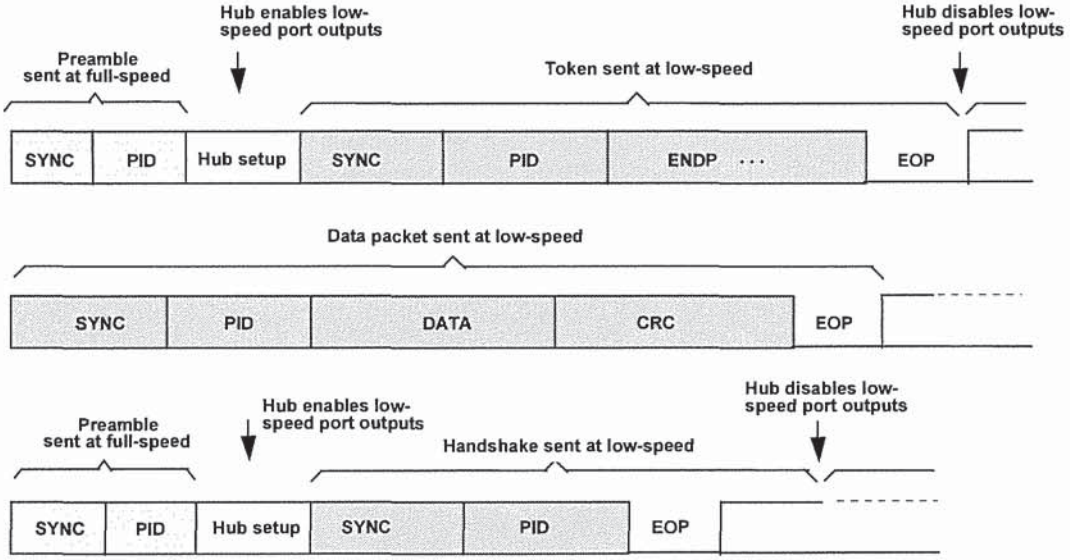


Figure 8-48. Low-speed Transaction

All downstream packets transmitted to low-speed devices within a full-/low-speed signaling environment require a preamble. Preambles are never used in a high-speed signaling environment. The preamble consists of a SYNC followed by a PRE PID, both sent at full-speed. Hubs must comprehend the PRE PID; all other USB devices may ignore it and treat it as undefined. At the end of the preamble PID, the host (or TT) drives the bus to the Idle state for at least one full-speed bit time. This Idle period on the bus is termed the hub setup interval and lasts for at least four full-speed bit times. During this hub setup interval, hubs must drive their full-speed and low-speed ports to their respective Idle states. Hubs must be ready to repeat low-speed signaling on low-speed ports before the end of the hub setup interval. Low-speed connectivity rules are summarized below:

1. Low-speed devices are identified during the connection process, and the hub ports to which they are connected are identified as low-speed.
2. All downstream low-speed packets must be prefaced with a preamble (sent at full-speed), which turns on the output buffers on low-speed hub ports.

3. Low-speed hub port output buffers are turned off upon receipt of EOP and are not turned on again until a preamble PID is detected.
4. Upstream connectivity is not affected by whether a hub port is full- or low-speed.

Low-speed signaling begins with the host (or TT) issuing SYNC at low-speed, followed by the remainder of the packet. The end of the packet is identified by an End-of-Packet (EOP), at which time all hubs tear down connectivity and disable any ports to which low-speed devices are connected. Hubs do not switch ports for upstream signaling; low-speed ports remain enabled in the upstream direction for both low-speed and full-speed signaling.

Low-speed and full-speed transactions maintain a high degree of protocol commonality. However, low-speed signaling does have certain limitations which include:

- Data payload is limited to eight bytes, maximum.
- Only interrupt and control types of transfers are supported.
- The SOF packet is not received by low-speed devices.

8.7 Error Detection and Recovery

The USB permits reliable end-to-end communication in the presence of errors on the physical signaling layer. This includes the ability to reliably detect the vast majority of possible errors and to recover from errors on a transaction-type basis. Control transactions, for example, require a high degree of data reliability; they support end-to-end data integrity using error detection and retry. Isochronous transactions, by virtue of their bandwidth and latency requirements, do not permit retries and must tolerate a higher incidence of uncorrected errors.

8.7.1 Packet Error Categories

The USB employs three error detection mechanisms: bit stuff violations, PID check bits, and CRCs. Bit stuff violations are defined in Section 7.1.9. PID errors are defined in Section 8.3.1. CRC errors are defined in Section 8.3.5.

With the exception of the SOF token, any packet that is received corrupted causes the receiver to ignore it and discard any data or other field information that came with the packet. Table 8-8 lists error detection mechanisms, the types of packets to which they apply, and the appropriate packet receiver response.

Table 8-8. Packet Error Types

Field	Error	Action
PID	PID Check, Bit Stuff	Ignore packet
Address	Bit Stuff, Address CRC	Ignore token
Frame Number	Bit Stuff, Frame Number CRC	Ignore Frame Number field
Data	Bit Stuff, Data CRC	Discard data

8.7.2 Bus Turn-around Timing

Neither the device nor the host will send an indication that a received packet had an error. This absence of positive acknowledgement is considered to be the indication that there was an error. As a consequence of this method of error reporting, the host and USB function need to keep track of how much time has elapsed from when the transmitter completes sending a packet until it begins to receive a response packet. This time is referred to as the bus turn-around time. Devices and hosts require turn-around timers to measure this time.

For full-/low-speed transactions, the timer starts counting on the SE0-to-'J' transition of the EOP strobe and stops counting when the Idle-to-'K' SOP transition is detected. For high-speed transactions, the timer starts counting when the data lines return to the squelch level and stops counting when the data lines leave the squelch level.

The device bus turn-around time is defined by the worst case round trip delay plus the maximum device response delay (refer to Sections 7.1.18 and 7.1.19 for specific bus turn-around times). If a response is not received within this worst case timeout, then the transmitter considers that the packet transmission has failed.

Timeout is used and interpreted as a transaction error condition for many transfer types. If the host wishes to indicate an error condition for a transaction via a timeout, it must wait the full bus turn-around time before issuing the next token to ensure that all downstream devices have timed out.

As shown in Figure 8-49, the device uses its bus turn-around timer between token and data or data and handshake phases. The host uses its timer between data and handshake or token and data phases.

If the host receives a corrupted data packet, it may require additional wait time before sending out the next token. This additional wait interval guarantees that the host properly handles false EOPs.

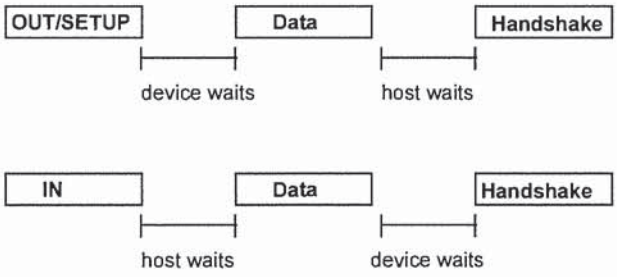


Figure 8-49. Bus Turn-around Timer Usage

8.7.3 False EOPs

False EOPs must be handled in a manner which guarantees that the packet currently in progress completes before the host or any other device attempts to transmit a new packet. If such an event were to occur, it would constitute a bus collision and have the ability to corrupt up to two consecutive transactions. Detection of false EOP relies upon the fact that a packet into which a false EOP has been inserted will appear as a truncated packet with a CRC failure. (The last 16 bits of the data packet will have a very low probability of appearing to be a correct CRC.)

The host and devices handle false EOP situations differently. When a device receives a corrupted data packet, it issues no response and waits for the host to send the next token. This scheme guarantees that the device will not attempt to return a handshake while the host may still be transmitting a data packet. If a false EOP has occurred, the host data packet will eventually end, and the device will be able to detect the next token. If a device issues a data packet that gets corrupted with a false EOP, the host will ignore the

packet and not issue the handshake. The device, expecting to see a handshake from the host, will timeout the transaction.

If the host receives a corrupted full-/low-speed data packet, it assumes that a false EOP may have occurred and waits for 16 bit times to see if there is any subsequent upstream traffic. If no bus transitions are detected within the 16 bit interval and the bus remains in the Idle state, the host may issue the next token.

Otherwise, the host waits for the device to finish sending the remainder of its full-/low-speed packet. Waiting 16 bit times guarantees two conditions:

- The first condition is to make sure that the device has finished sending its packet. This is guaranteed by a timeout interval (with no bus transitions) greater than the worst case six-bit time bit stuff interval.
- The second condition is that the transmitting device's bus turn-around timer must be guaranteed to expire.

Note that the timeout interval is transaction speed sensitive. For full-speed transactions, the host must wait full-speed bit times; for low-speed transactions, it must wait low-speed bit times.

If the host receives a corrupted high-speed data packet, it ignores any data until the data lines return to the squelch level before issuing the next token. For high-speed transactions, the host does not need to wait additional time (beyond the normal inter-transaction gap time) after the data lines return to the squelch level.

If the host receives a data packet with a valid CRC, it assumes that the packet is complete and requires no additional delay (beyond normal inter-transaction gap time) in issuing the next token.

8.7.4 Babble and Loss of Activity Recovery

The USB must be able to detect and recover from conditions which leave it waiting indefinitely for a full-/low-speed EOP or which leave the bus in something other than the Idle state at the end of a (micro)frame.

- Full-/low-speed loss of activity (LOA) is characterized by an SOP followed by lack of bus activity (bus remains driven to a 'J' or 'K') and no EOP at the end of a frame.
- Full-/low-speed babble is characterized by an SOP followed by the presence of bus activity past the end of a frame.
- High-speed babble/LOA is characterized by the data lines being at an unsquelched level at the end of a microframe.

LOA and babble have the potential to either deadlock the bus or delay the beginning of the next (micro)frame. Neither condition is acceptable, and both must be prevented from occurring. As the USB component responsible for controlling connectivity, hubs are responsible for babble/LOA detection and recovery. All USB devices that fail to complete their transmission at the end of a (micro)frame are prevented from transmitting past a (micro)frame's end by having the nearest hub disable the port to which the offending device is attached. Details of the hub babble/LOA recovery mechanism appear in Section 11.2.5.

Chapter 9
USB Device Framework

A USB device may be divided into three layers:

- The bottom layer is a bus interface that transmits and receives packets.
- The middle layer handles routing data between the bus interface and various endpoints on the device. An endpoint is the ultimate consumer or provider of data. It may be thought of as a source or sink for data.
- The top layer is the functionality provided by the serial bus device, for instance, a mouse or ISDN interface.

This chapter describes the common attributes and operations of the middle layer of a USB device. These attributes and operations are used by the function-specific portions of the device to communicate through the bus interface and ultimately with the host.

9.1 USB Device States

A USB device has several possible states. Some of these states are visible to the USB and the host, while others are internal to the USB device. This section describes those states.

9.1.1 Visible Device States

This section describes USB device states that are externally visible (see Figure 9-1). Table 9-1 summarizes the visible device states.

Note: USB devices perform a reset operation in response to reset signaling on the upstream facing port. When reset signaling has completed, the USB device is reset.

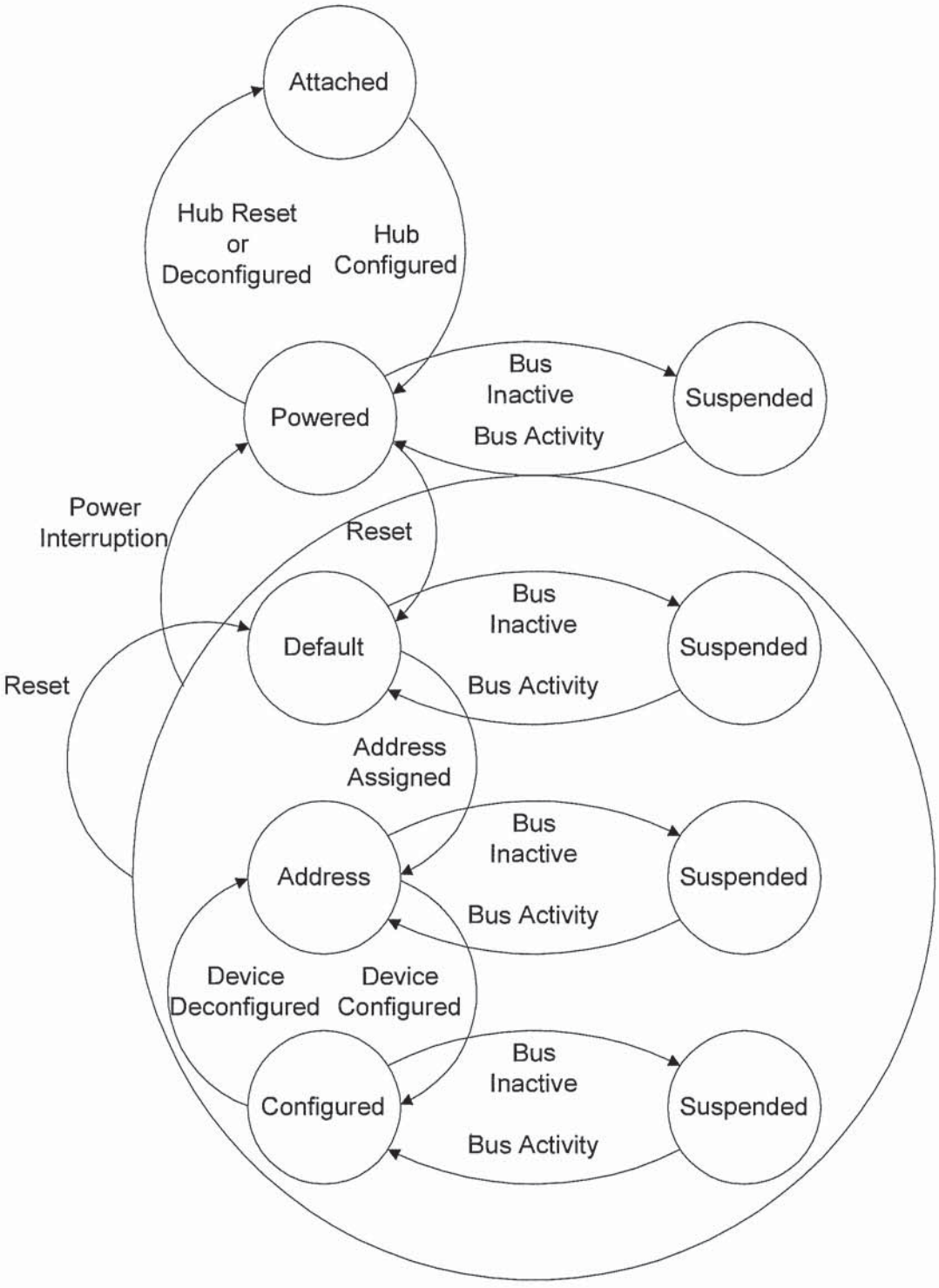


Figure 9-1. Device State Diagram

Table 9-1. Visible Device States

Attached	Powered	Default	Address	Configured	Suspended	State
No	--	--	--	--	--	Device is not attached to the USB. Other attributes are not significant.
Yes	No	--	--	--	--	Device is attached to the USB, but is not powered. Other attributes are not significant.
Yes	Yes	No	--	--	--	Device is attached to the USB and powered, but has not been reset.
Yes	Yes	Yes	No	--	--	Device is attached to the USB and powered and has been reset, but has not been assigned a unique address. Device responds at the default address.
Yes	Yes	Yes	Yes	No	--	Device is attached to the USB, powered, has been reset, and a unique device address has been assigned. Device is not configured.
Yes	Yes	Yes	Yes	Yes	No	Device is attached to the USB, powered, has been reset, has a unique address, is configured, and is not suspended. The host may now use the function provided by the device.
Yes	Yes	--	--	--	Yes	Device is, at minimum, attached to the USB and is powered and has not seen bus activity for 3 ms. It may also have a unique address and be configured for use. However, because the device is suspended, the host may not use the device's function.

9.1.1.1 Attached

A USB device may be attached or detached from the USB. The state of a USB device when it is detached from the USB is not defined by this specification. This specification only addresses required operations and attributes once the device is attached.

9.1.1.2 Powered

USB devices may obtain power from an external source and/or from the USB through the hub to which they are attached. Externally powered USB devices are termed self-powered. Although self-powered devices may already be powered before they are attached to the USB, they are not considered to be in the Powered state until they are attached to the USB and VBUS is applied to the device.

A device may support both self-powered and bus-powered configurations. Some device configurations support either power source. Other device configurations may be available only if the device is self-powered. Devices report their power source capability through the configuration descriptor. The current power source is reported as part of a device's status. Devices may change their power source at any time, e.g., from self- to bus-powered. If a configuration is capable of supporting both power modes, the power maximum reported for that configuration is the maximum the device will draw from VBUS in either mode. The device must observe this maximum, regardless of its mode. If a configuration supports only one power mode and the power source of the device changes, the device will lose its current configuration and address and return to the Powered state. If a device is self-powered and its current configuration requires more than 100 mA, then if the device switches to being bus-powered, it must return to the Address state. Self-powered hubs that use VBUS to power the Hub Controller are allowed to remain in the Configured state if local power is lost. Refer to Section 11.13 for details.

A hub port must be powered in order to detect port status changes, including attach and detach. Bus-powered hubs do not provide any downstream power until they are configured, at which point they will provide power as allowed by their configuration and power source. A USB device must be able to be addressed within a specified time period from when power is initially applied (refer to Chapter 7). After an attachment to a port has been detected, the host may enable the port, which will also reset the device attached to the port.

9.1.1.3 Default

After the device has been powered, it must not respond to any bus transactions until it has received a reset from the bus. After receiving a reset, the device is then addressable at the default address.

When the reset process is complete, the USB device is operating at the correct speed (i.e., low-/full-/high-speed). The speed selection for low- and full-speed is determined by the device termination resistors. A device that is capable of high-speed operation determines whether it will operate at high-speed as a part of the reset process (see Chapter 7 for more details).

A device capable of high-speed operation must reset successfully at full-speed when in an electrical environment that is operating at full-speed. After the device is successfully reset, the device must also respond successfully to device and configuration descriptor requests and return appropriate information. The device may or may not be able to support its intended functionality when operating at full-speed.

9.1.1.4 Address

All USB devices use the default address when initially powered or after the device has been reset. Each USB device is assigned a unique address by the host after attachment or after reset. A USB device maintains its assigned address while suspended.

A USB device responds to requests on its default pipe whether the device is currently assigned a unique address or is using the default address.

9.1.1.5 Configured

Before a USB device's function may be used, the device must be configured. From the device's perspective, configuration involves correctly processing a SetConfiguration() request with a non-zero configuration value. Configuring a device or changing an alternate setting causes all of the status and configuration values associated with endpoints in the affected interfaces to be set to their default values. This includes setting the data toggle of any endpoint using data toggles to the value DATA0.

9.1.1.6 Suspended

In order to conserve power, USB devices automatically enter the Suspended state when the device has observed no bus traffic for a specified period (refer to Chapter 7). When suspended, the USB device maintains any internal status, including its address and configuration.

All devices must suspend if bus activity has not been observed for the length of time specified in Chapter 7. Attached devices must be prepared to suspend at any time they are powered, whether they have been assigned a non-default address or are configured. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the Suspended state when the hub port it is attached to is disabled. This is referred to as selective suspend.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If a USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

9.1.2 Bus Enumeration

When a USB device is attached to or removed from the USB, the host uses a process known as bus enumeration to identify and manage the device state changes necessary. When a USB device is attached to a powered port, the following actions are taken:

1. The hub to which the USB device is now attached informs the host of the event via a reply on its status change pipe (refer to Section 11.12.3 for more information). At this point, the USB device is in the Powered state and the port to which it is attached is disabled.
2. The host determines the exact nature of the change by querying the hub.
3. Now that the host knows the port to which the new device has been attached, the host then waits for at least 100 ms to allow completion of an insertion process and for power at the device to become stable. The host then issues a port enable and reset command to that port. Refer to Section 7.1.7.5 for sequence of events and timings of connection through device reset.
4. The hub performs the required reset processing for that port (see Section 11.5.1.5). When the reset signal is released, the port has been enabled. The USB device is now in the Default state and can draw no more than 100 mA from VBUS. All of its registers and state have been reset and it answers to the default address.
5. The host assigns a unique address to the USB device, moving the device to the Address state.
6. Before the USB device receives a unique address, its Default Control Pipe is still accessible via the default address. The host reads the device descriptor to determine what actual maximum data payload size this USB device's default pipe can use.
7. The host reads the configuration information from the device by reading each configuration zero to n-1, where n is the number of configurations. This process may take several milliseconds to complete.

8. Based on the configuration information and how the USB device will be used, the host assigns a configuration value to the device. The device is now in the Configured state and all of the endpoints in this configuration have taken on their described characteristics. The USB device may now draw the amount of VBUS power described in its descriptor for the selected configuration. From the device's point of view, it is now ready for use.

When the USB device is removed, the hub again sends a notification to the host. Detaching a device disables the port to which it had been attached. Upon receiving the detach notification, the host will update its local topological information.

9.2 Generic USB Device Operations

All USB devices support a common set of operations. This section describes those operations.

9.2.1 Dynamic Attachment and Removal

USB devices may be attached and removed at any time. The hub that provides the attachment point or port is responsible for reporting any change in the state of the port.

The host enables the hub port where the device is attached upon detection of an attachment, which also has the effect of resetting the device. A reset USB device has the following characteristics:

- Responds to the default USB address
- Is not configured
- Is not initially suspended

When a device is removed from a hub port, the hub disables the port where the device was attached and notifies the host of the removal.

9.2.2 Address Assignment

When a USB device is attached, the host is responsible for assigning a unique address to the device. This is done after the device has been reset by the host, and the hub port where the device is attached has been enabled.

9.2.3 Configuration

A USB device must be configured before its function(s) may be used. The host is responsible for configuring a USB device. The host typically requests configuration information from the USB device to determine the device's capabilities.

As part of the configuration process, the host sets the device configuration and, where necessary, selects the appropriate alternate settings for the interfaces.

Within a single configuration, a device may support multiple interfaces. An interface is a related set of endpoints that present a single feature or function of the device to the host. The protocol used to communicate with this related set of endpoints and the purpose of each endpoint within the interface may be specified as part of a device class or vendor-specific definition.

In addition, an interface within a configuration may have alternate settings that redefine the number or characteristics of the associated endpoints. If this is the case, the device must support the GetInterface() request to report the current alternate setting for the specified interface and SetInterface() request to select the alternate setting for the specified interface.

Within each configuration, each interface descriptor contains fields that identify the interface number and the alternate setting. Interfaces are numbered from zero to one less than the number of concurrent interfaces supported by the configuration. Alternate settings range from zero to one less than the number of alternate

settings for a specific interface. The default setting when a device is initially configured is alternate setting zero.

In support of adaptive device drivers that are capable of managing a related group of USB devices, the device and interface descriptors contain *Class*, *SubClass*, and *Protocol* fields. These fields are used to identify the function(s) provided by a USB device and the protocols used to communicate with the function(s) on the device. A class code is assigned to a group of related devices that has been characterized as a part of a USB Class Specification. A class of devices may be further subdivided into subclasses, and, within a class or subclass, a protocol code may define how the Host Software communicates with the device.

Note: The assignment of class, subclass, and protocol codes must be coordinated but is beyond the scope of this specification.

9.2.4 Data Transfer

Data may be transferred between a USB device endpoint and the host in one of four ways. Refer to Chapter 5 for the definition of the four types of transfers. An endpoint number may be used for different types of data transfers in different alternate settings. However, once an alternate setting is selected (including the default setting of an interface), a USB device endpoint uses only one data transfer method until a different alternate setting is selected.

9.2.5 Power Management

Power management on USB devices involves the issues described in the following sections.

9.2.5.1 Power Budgeting

USB bus power is a limited resource. During device enumeration, a host evaluates a device's power requirements. If the power requirements of a particular configuration exceed the power available to the device, Host Software shall not select that configuration.

USB devices shall limit the power they consume from VBUS to one unit load or less until configured. Suspended devices, whether configured or not, shall limit their bus power consumption as defined in Chapter 7. Depending on the power capabilities of the port to which the device is attached, a USB device may be able to draw up to five unit loads from VBUS after configuration.

9.2.5.2 Remote Wakeup

Remote wakeup allows a suspended USB device to signal a host that may also be suspended. This notifies the host that it should resume from its suspended mode, if necessary, and service the external event that triggered the suspended USB device to signal the host. A USB device reports its ability to support remote wakeup in a configuration descriptor. If a device supports remote wakeup, it must also allow the capability to be enabled and disabled using the standard USB requests.

Remote wakeup is accomplished using electrical signaling described in Section 7.1.7.7.

9.2.6 Request Processing

With the exception of SetAddress() requests (see Section 9.4.6), a device may begin processing of a request as soon as the device returns the ACK following the Setup. The device is expected to "complete" processing of the request before it allows the Status stage to complete successfully. Some requests initiate operations that take many milliseconds to complete. For requests such as this, the device class is required to define a method other than Status stage completion to indicate that the operation has completed. For example, a reset on a hub port takes at least 10 ms to complete. The SetPortFeature(PORT_RESET) (see Chapter 11) request "completes" when the reset on the port is initiated. Completion of the reset operation is

signaled when the port's status change is set to indicate that the port is now enabled. This technique prevents the host from having to constantly poll for a completion when it is known that the request will take a relatively long period of time.

9.2.6.1 Request Processing Timing

All devices are expected to handle requests in a timely manner. USB sets an upper limit of 5 seconds as the upper limit for any command to be processed. This limit is not applicable in all instances. The limitations are described in the following sections. It should be noted that the limitations given below are intended to encompass a wide range of implementations. If all devices in a USB system used the maximum allotted time for request processing, the user experience would suffer. For this reason, implementations should strive to complete requests in times that are as short as possible.

9.2.6.2 Reset/Resume Recovery Time

After a port is reset or resumed, the USB System Software is expected to provide a "recovery" interval of 10 ms before the device attached to the port is expected to respond to data transfers. The device may ignore any data transfers during the recovery interval.

After the end of the recovery interval (measured from the end of the reset or the end of the EOP at the end of the resume signaling), the device must accept data transfers at any time.

9.2.6.3 Set Address Processing

After the reset/resume recovery interval, if a device receives a SetAddress() request, the device must be able to complete processing of the request and be able to successfully complete the Status stage of the request within 50 ms. In the case of the SetAddress() request, the Status stage successfully completes when the device sends the zero-length Status packet or when the device sees the ACK in response to the Status stage data packet.

After successful completion of the Status stage, the device is allowed a SetAddress() recovery interval of 2 ms. At the end of this interval, the device must be able to accept Setup packets addressed to the new address. Also, at the end of the recovery interval, the device must not respond to tokens sent to the old address (unless, of course, the old and new address is the same).

9.2.6.4 Standard Device Requests

For standard device requests that require no Data stage, a device must be able to complete the request and be able to successfully complete the Status stage of the request within 50 ms of receipt of the request. This limitation applies to requests to the device, interface, or endpoint.

For standard device requests that require data stage transfer to the host, the device must be able to return the first data packet to the host within 500 ms of receipt of the request. For subsequent data packets, if any, the device must be able to return them within 500 ms of successful completion of the transmission of the previous packet. The device must then be able to successfully complete the status stage within 50 ms after returning the last data packet.

For standard device requests that require a data stage transfer to the device, the 5-second limit applies. This means that the device must be capable of accepting all data packets from the host and successfully completing the Status stage if the host provides the data at the maximum rate at which the device can accept it. Delays between packets introduced by the host add to the time allowed for the device to complete the request.

9.2.6.5 Class-specific Requests

Unless specifically exempted in the class document, all class-specific requests must meet the timing limitations for standard device requests. If a class document provides an exemption, the exemption may only be specified on a request-by-request basis.

A class document may require that a device respond more quickly than is specified in this section. Faster response may be required for standard and class-specific requests.

9.2.6.6 Speed Dependent Descriptors

A device capable of operation at high-speed can operate in either full- or high-speed. The device always knows its operational speed due to having to manage its transceivers correctly as part of reset processing (See Chapter 7 for more details on reset). A device also operates at a single speed after completing the reset sequence. In particular, there is no speed switch during normal operation. However, a high-speed capable device may have configurations that are speed dependent. That is, it may have some configurations that are only possible when operating at high-speed or some that are only possible when operating at full-speed. High-speed capable devices must support reporting their speed dependent configurations.

A high-speed capable device responds with descriptor information that is valid for the current operating speed. For example, when a device is asked for configuration descriptors, it only returns those for the current operating speed (e.g., full speed). However, there must be a way to determine the capabilities for both high- and full-speed operation.

Two descriptors allow a high-speed capable device to report configuration information about the other operating speed. The two descriptors are: the (other_speed) device_qualifier descriptor and the other_speed_configuration descriptor. These two descriptors are retrieved by the host by using the GetDescriptor request with the corresponding descriptor type values.

Note: These descriptors are not retrieved unless the host explicitly issues the corresponding GetDescriptor requests. If these two requests are not issued, the device would simply appear to be a single speed device.

Devices that are high-speed capable must set the version number in the *bcdUSB* field of their descriptors to 0200H. This indicates that such devices support the other_speed requests defined by USB 2.0. A device with descriptor version numbers less than 0200H should cause a Request Error response (see next section) if it receives these other_speed requests. A USB 1.x device (i.e., one with a device descriptor version less than 0200H) should not be issued the other_speed requests.

9.2.7 Request Error

When a request is received by a device that is not defined for the device, is inappropriate for the current setting of the device, or has values that are not compatible with the request, then a Request Error exists. The device deals with the Request Error by returning a STALL PID in response to the next Data stage transaction or in the Status stage of the message. It is preferred that the STALL PID be returned at the next Data stage transaction, as this avoids unnecessary bus activity.

9.3 USB Device Requests

All USB devices respond to requests from the host on the device's Default Control Pipe. These requests are made using control transfers. The request and the request's parameters are sent to the device in the Setup packet. The host is responsible for establishing the values passed in the fields listed in Table 9-2. Every Setup packet has eight bytes.

Table 9-2. Format of Setup Data

Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved
1	*bRequest*	1	Value	Specific request (refer to Table 9-3)
2	*wValue*	2	Value	Word-sized field that varies according to request
4	*wIndex*	2	Index or Offset	Word-sized field that varies according to request; typically used to pass an index or offset
6	*wLength*	2	Count	Number of bytes to transfer if there is a Data stage

9.3.1 bmRequestType

This bitmapped field identifies the characteristics of the specific request. In particular, this field identifies the direction of data transfer in the second phase of the control transfer. The state of the *Direction* bit is ignored if the *wLength* field is zero, signifying there is no Data stage.

The USB Specification defines a series of standard requests that all devices must support. These are enumerated in Table 9-3. In addition, a device class may define additional requests. A device vendor may also define requests supported by the device.

Requests may be directed to the device, an interface on the device, or a specific endpoint on a device. This field also specifies the intended recipient of the request. When an interface or endpoint is specified, the *wIndex* field identifies the interface or endpoint.

9.3.2 bRequest

This field specifies the particular request. The *Type* bits in the *bmRequestType* field modify the meaning of this field. This specification defines values for the *bRequest* field only when the bits are reset to zero, indicating a standard request (refer to Table 9-3).

9.3.3 wValue

The contents of this field vary according to the request. It is used to pass a parameter to the device, specific to the request.

9.3.4 wIndex

The contents of this field vary according to the request. It is used to pass a parameter to the device, specific to the request.

The *wIndex* field is often used in requests to specify an endpoint or an interface. Figure 9-2 shows the format of *wIndex* when it is used to specify an endpoint.

D7	D6	D5	D4	D3	D2	D1	D0
Direction	Reserved (Reset to zero)			Endpoint Number			
D15	D14	D13	D12	D11	D10	D9	D8
Reserved (Reset to zero)							

Figure 9-2. wIndex Format when Specifying an Endpoint

The *Direction* bit is set to zero to indicate the OUT endpoint with the specified *Endpoint Number* and to one to indicate the IN endpoint. In the case of a control pipe, the request should have the *Direction* bit set to zero but the device may accept either value of the *Direction* bit.

Figure 9-3 shows the format of *wIndex* when it is used to specify an interface.

D7	D6	D5	D4	D3	D2	D1	D0
Interface Number							
D15	D14	D13	D12	D11	D10	D9	D8
Reserved (Reset to zero)							

Figure 9-3. wIndex Format when Specifying an Interface

9.3.5 wLength

This field specifies the length of the data transferred during the second phase of the control transfer. The direction of data transfer (host-to-device or device-to-host) is indicated by the *Direction* bit of the *bmRequestType* field. If this field is zero, there is no data transfer phase.

On an input request, a device must never return more data than is indicated by the *wLength* value; it may return less. On an output request, *wLength* will always indicate the exact amount of data to be sent by the host. Device behavior is undefined if the host should send more data than is specified in *wLength*.

9.4 Standard Device Requests

This section describes the standard device requests defined for all USB devices. Table 9-3 outlines the standard device requests, while Table 9-4 and Table 9-5 give the standard request codes and descriptor types, respectively.

USB devices must respond to standard device requests, even if the device has not yet been assigned an address or has not been configured.

Table 9-3. Standard Device Requests

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
10000000B	GET_CONFIGURATION	Zero	Zero	One	Configuration Value
10000000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
10000001B	GET_INTERFACE	Zero	Interface	One	Alternate Interface
10000000B 10000001B 10000010B	GET_STATUS	Zero	Zero Interface Endpoint	Two	Device, Interface, or Endpoint Status
00000000B	SET_ADDRESS	Device Address	Zero	Zero	None
00000000B	SET_CONFIGURATION	Configuration Value	Zero	Zero	None
00000000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
00000000B 00000001B 00000010B	SET_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
00000001B	SET_INTERFACE	Alternate Setting	Interface	Zero	None
10000010B	SYNCH_FRAME	Zero	Endpoint	Two	Frame Number

Table 9-4. Standard Request Codes

bRequest	Value
GET_STATUS	0
CLEAR_FEATURE	1
Reserved for future use	2
SET_FEATURE	3
Reserved for future use	4
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12

Table 9-5. Descriptor Types

Descriptor Types	Value
DEVICE	1
CONFIGURATION	2
STRING	3
INTERFACE	4
ENDPOINT	5
DEVICE_QUALIFIER	6
OTHER_SPEED_CONFIGURATION	7
INTERFACE_POWER[1]	8

[1] The INTERFACE_POWER descriptor is defined in the current revision of the *USB Interface Power Management Specification*.