Chapter 11
Hub Specification

This chapter describes the architectural requirements for the USB hub. It contains a description of the two principal sub-blocks: the hub repeater and the hub controller. The chapter also describes the hubs operation for error recovery, reset, and suspend/resume. The second half of the chapter defines hub request behavior and hub descriptors.

The hub specification supplies sufficient information to permit an implementer to design a USB hub which conforms to the USB specification.

11.1 Overview

Hubs provide the electrical interface between USB devices and the host and are directly responsible for supporting many of the attributes that make USB user friendly and hide its complexity from the user. Listed below are the major aspects of USB functionality that hubs must support:

- Connectivity behavior
- Power management
- Device connect/disconnect detection
- Bus fault detection and recovery
- Full/Low speed device support

A hub consists of two components, the hub repeater and the hub controller. The repeater is responsible for connectivity setup and tear-down. It also supports exception handling such as bus fault detection and recovery and connect/disconnect detect. The hub controller provides the mechanism for host to hub communication. Hub specific status and control commands permit the host to configure a hub and to monitor and control its individual downstream ports.

11.2 Device Characteristics

11.2.1 Hub Architecture

Figure 11-1 shows a hub and the locations of its root and downstream ports. A hub consists of a repeater section and a hub controller section. The repeater is responsible for managing connectivity on a per packet basis, while the hub controller provides status and control and permits host access to the hub.

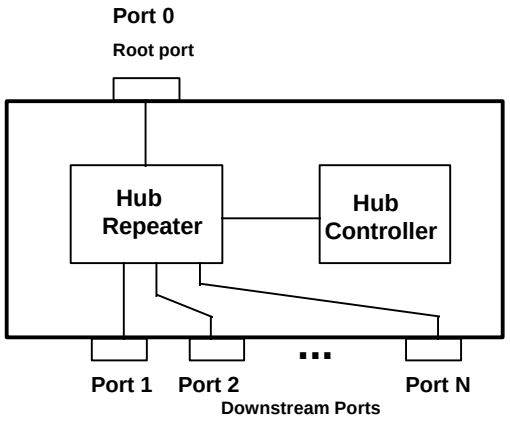

Figure 11-1 Hub Architecture

11.2.2 Hub Connectivity

Hubs display differing connectivity behavior depending on whether they are propagating packet traffic or resume signaling, or are in the idle state.

11.2.2.1 Packet Signaling Connectivity

The hub repeater contains one port that must always connect in the upstream direction (referred to as the root port or upstream port) and one or more downstream ports. Upstream connectivity is defined as being towards the host, and downstream connectivity is defined as being towards a device. Figure 11-2. Hub Signaling Connectivity shows the packet signaling connectivity behavior for hubs in the upstream and downstream directions. A hub also has an idle state during which the hub makes no connectivity. When in the idle state all of the hub's ports are in the receive mode waiting for the start of the next packet.

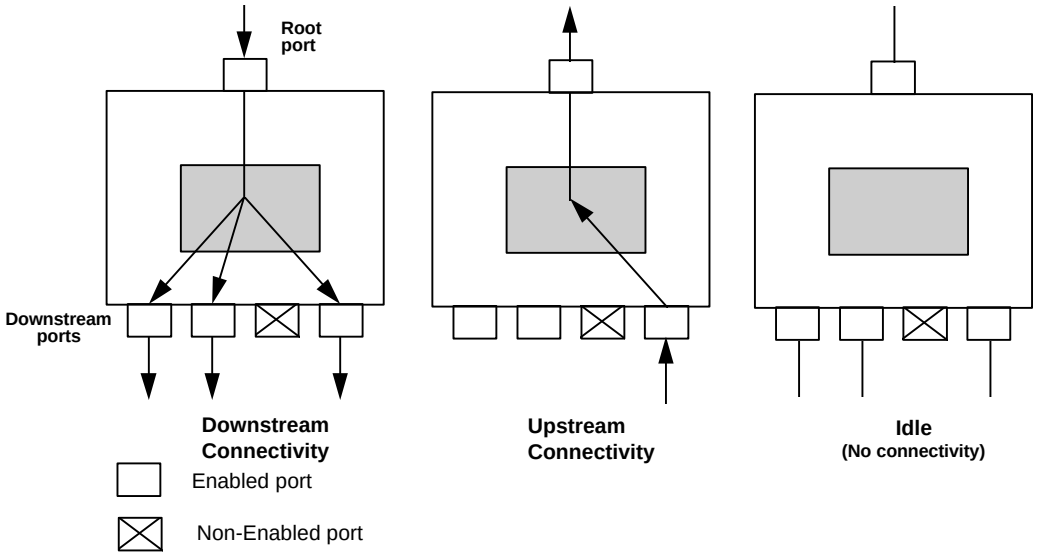

Figure 11-2. Hub Signaling Connectivity

If a downstream hub port is enabled (i.e., in a state where it can propagate signaling through the hub) and the hub detects an SOP on that port, connectivity is established in an upstream direction to the root port of that hub, but not to any other downstream ports. This means that when a device or a hub transmits a packet upstream, only those hubs in line between the transmitting device and the host will see the packet. After an SOP on a downstream port is detected, connectivity from all other downstream ports is blocked. This guarantees that hub connectivity will not be modified until the next EOP is detected or until the hub times out at the end of the frame.

In the downstream direction, hubs operate in a broadcast mode. When a hub detects an SOP on its root port, it establishes connectivity to all enabled downstream ports. If a port is not enabled, it does not receive any packet traffic activity from the root port and does not propagate packet signaling downstream.

11.2.2.2 Resume Connectivity

Hubs exhibit differing connectivity behaviors for upstream and downstream directed resume signaling. A hub which is in the suspend state reflects resume signaling from its root port to all of its enabled downstream ports. Figure 11-3 illustrates hub upstream and downstream resume connectivity.

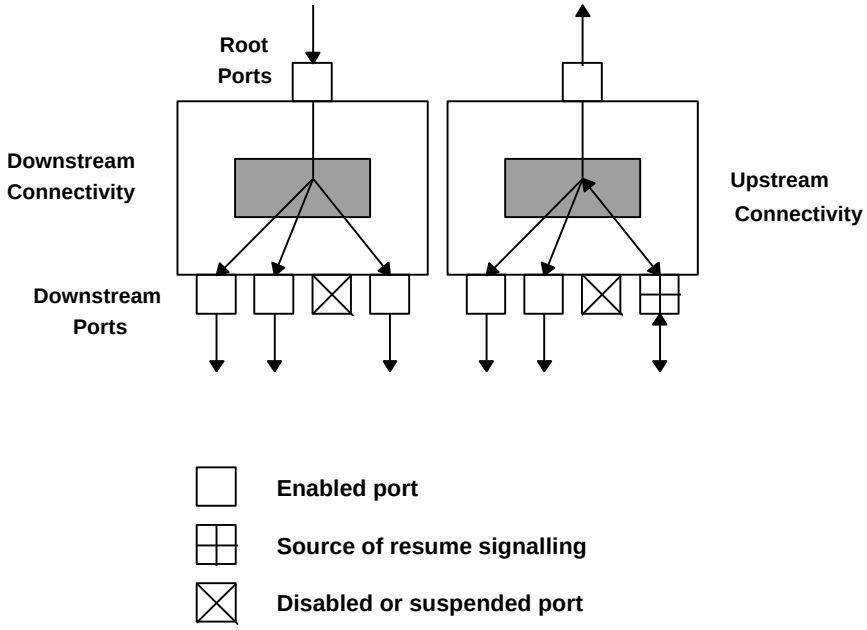

Figure 11-3. Resume Connectivity

If a hub is in the suspend state and detects resume signaling from a selectively suspended or enabled downstream port, the hub reflects that signaling upstream to its root and also to all of its enabled downstream ports, including the port which initiated the resume sequence. Resume signaling is not reflected to disabled or other suspended ports. A detailed discussion of resume connectivity appears in section 11.5.

11.2.3 Hub Port States

A hub downstream port may be in one of four or five possible states and must comprehend such features as connect/disconnect detect, port enable/disable, suspend/resume, reset, and, optionally, power switching. Hubs must support independent port state machines on a per downstream port basis. Figure 11-4 illustrates the states for a non-power switched port, and Figure 11-5 shows the states for a port with power switching. Hub port states apply only to downstream ports.

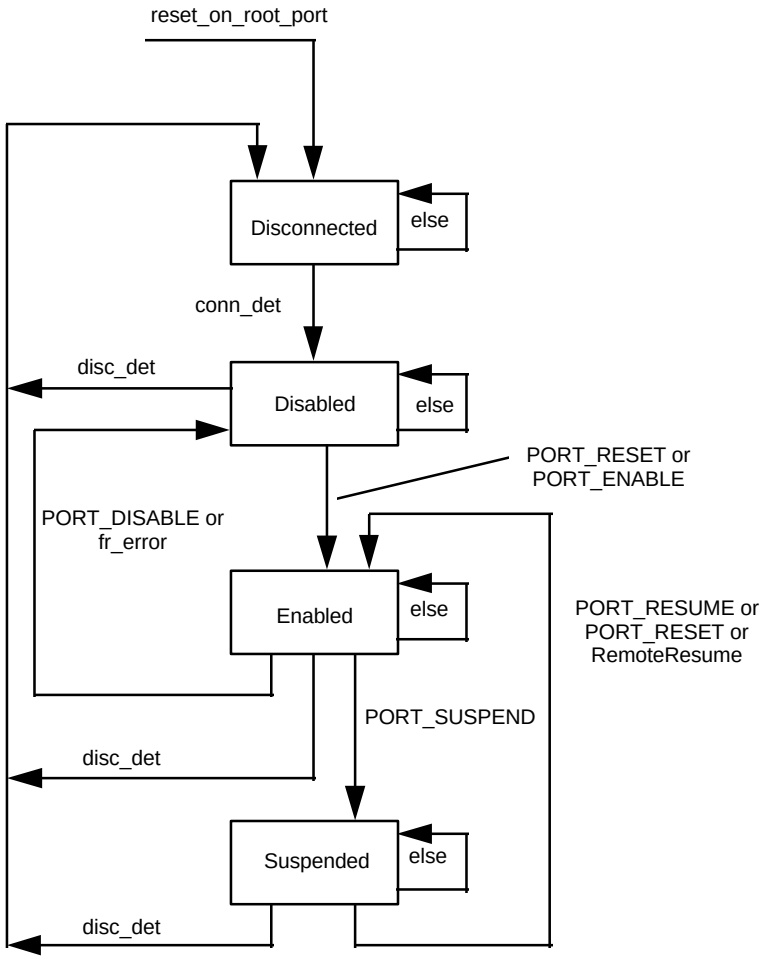

Figure 11-4. Non-power Switched Hub Port States

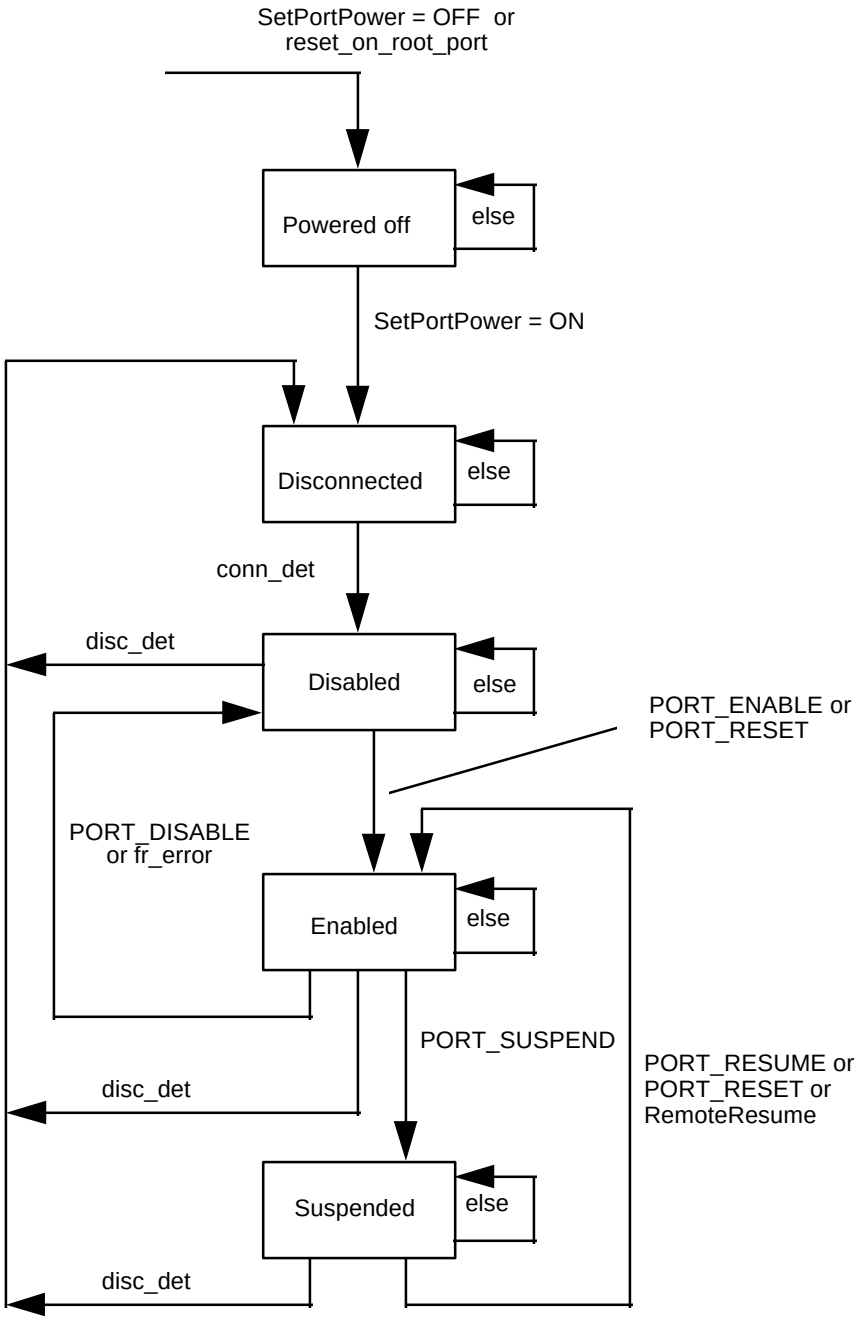

Figure 11-5. Power Switched Hub Port States

Each of the states is described below.

Powered off: This state is only supported for ports that have power switching as shown in Figure 11-5. A port transitions to its powered off state when the hub receives a ClearPortFeature(PORT_POWER) request, or when the hub detects root port reset signaling. A hub's downstream ports also transition to the powered off state when power is initially applied to a hub. A

powered off port supplies no power downstream, and its signal outputs buffers are placed in the Hi-Z state. A powered off port ignores all upstream directed bus activity on that port.

Disconnected: A downstream port of a hub supporting power switching transitions from the powered off to the disconnected state when power is applied to the port via a SetPortFeature(PORT_POWER) request. If a hub does not support power switching, it transitions to the disconnected state upon power up or upon receipt of a root port reset. In the disconnected state, a port cannot propagate any signaling in either the upstream or downstream direction. However, the port can detect a connect event (conn_det), which sets a status field in the hub controller and causes the port state to transition to the disabled state.

Conn_det is asserted if a downstream port is in the disconnect state and detects at least 2.5 μs of continuous non SE0 signaling on the bus. When conn_det is first asserted, the idle bus state can be driven from either a low or full speed device, and may be either a DIFF0 or a DIFF1. The hub automatically determines the device type (low or full speed) by determining whether D+ or D- is pulled high. Device speed determination is completed before the hub transitions to the disabled state.

Disabled: A port transitions to the disabled state from the disconnected state when it detects a connect event (conn_det). This requires that the port first be power switched on if the hub supports power switching. A port in the disabled state can only propagate downstream directed signaling arising from a SetPortFeature(RESET) request; at all other times, the port's output buffers are in the Hi-Z state. In the upstream direction, a port in the disabled state does not propagate any signaling through the hub to the root port when the hub is awake. However, certain events, such as disconnect will cause upstream resume signaling to be propagated to the root port if the hub is in the suspend state and its DEVICE_REMOTE_WAKEUP feature is enabled. A disconnect event (disc_det) will cause the port to return to the disconnect state and will set a status field in the hub controller. Disc_det is asserted whenever the port detects at least 2.5 μs of continuous SE0 when the port is not propagating downstream traffic. Before a disconnect event can be timed the suspended hub must first wake up.

Enabled: A port transitions to the enabled state upon receipt of a SetPortFeature(PORT_ENABLE) or a SetPortFeature(PORT_RESET) request. In the enabled state a full-speed port propagates all downstream signaling a low-speed port propagates downstream low-speed packet traffic when preceded by the preamble PID. An enabled port propagates all upstream signaling including full speed and low speed packets and resume signaling. A port transitions to the disabled state if it receives a ClearPortFeature(PORT_ENABLE) request or if a frame error (fr_error) occurs. The ClearPortFeature(PORT_ENABLE) request may be issued at any time by the host, and the hub must respond by immediately placing the port in the disabled state. An enabled port will transition to the disconnected state if a disconnect detect occurs.

Suspended: The hub selectively suspends all devices downstream of a port when it receives a SetPortFeature(PORT_SUSPEND) request. This request must not cause the now suspended port to stop propagating in the middle of a packet; i.e., the current packet must complete before the port enters the suspend state. A port displays different suspend connectivity behavior depending on whether the hub is awake or is itself suspended. If the hub is awake, no upstream or downstream connectivity can propagate through the port. However, if the hub is suspended, an idle to resume or an idle to SE0 transition (with hub feature DEVICE_REMOTE_WAKEUP enabled) on the port is reflected onto all other enabled ports (including the root port) as an idle to resume transition. This behavior makes it possible to suspend multiple hubs in series and still have a device at the bottom be able to wake up the entire bus. The port transitions from suspended to enabled state on commands from the host or on resume signaling from a device (remote wakeup). Remote wakeup and Selective resume are described in more detail in Section 11.5. The event RemoteResume indicates the end of the resume signaling used in the remote wakeup of a selectively suspended port.

If a hub is suspended and bus activity occurs on a suspended port, the hub first wakes up. The termination of a resume request to the port causes a status field to be set in the hub. In response, the

host polls the hub to read the status field change and determine on which port the resume occurred. The hub port transitions to the enabled state when the resume is complete. Details of the hub port signaling for selective resume are described in section 11.5.2**.**

A disconnect event to a suspended port causes the port state to transition to the disconnected state and sets the port status field to indicate that a disconnect has occurred. It is not possible to place a disconnected port directly into the suspend mode, since the port never exits the disconnected state.

Table 11-1 summarizes hub port behavior in different port states for different types of signaling. Hub behavior during resume signaling when the hub itself is in the suspend state constitutes a special case, and is discussed in Section 11.5.2.1.

Table 11-1. Port Behavior vs. Signaling

Signaling\State	Powered-off	Disconnected	Disabled	Enabled	Suspended
Reset on root port (hub with power switching)	Ignore	Go to **Powered off**	Go to **Powered off**	Go to **Powered off**	Go to **Powered off**
Reset on root port (hub without power switching)	N.A.	Ignore	Go to **Disconnected**	Go to **Disconnected**	Go to **Disconnected**
ClearPortFeature(PORT_POWER**)**(hub with power switching)	Ignore	Go to **Powered off**	Go to **Powered off**	Go to **Powered off**	Go to **Powered off**
SetPortFeature(PORT_POWER**)**(hub with power switching)	Go to **Disconnected**	Ignore	Ignore	Ignore	Ignore
SetPortFeature(PORT_RESET**) request**	Ignore	Ignore	issue reset, Go to **Enabled**	issue reset, Stay in **Enabled**	issue reset, Go to **Enabled**
SetPortFeature(PORT_ENABLE**)** request	Ignore	Ignore	Go to **Enabled**	Ignore	Ignore
ClearPortFeature(PORT_ENABLE**)** request	Ignore	Ignore	Ignore	Go to **Disabled**	Ignore
Frame error	N.A.	N.A.	N.A.	Go to **Disabled**	N.A.
RemoteResume	N.A.	N.A.	Ignore	N.A.	Go to **Enabled**
SetPortFeature(PORT_SUSPEND**)**request	Ignore	Ignore	Ignore	Go to **Suspended**	Ignore
ClearPortFeature(PORT_SUSPEND**)** request	Ignore	Ignore	Ignore	Ignore	issue resume, Go to **Enabled**
Disconnect detect	Ignore	Ignore	Go to **Disconnected**	Go to **Disconnected**	Go to **Disconnected**
Connect Event	Ignore	Go to **Disabled**	N.A.	N.A.	N.A.

11.2.3.1 Device Detach Detection

A hub is able to detect a detach event by means of a continuous SE0 persisting for at least 2.5 µs detected at a downstream port. In response to a detach event, the hub places the port into the Disconnected state and floats its output buffers to a Hi-Z. Device detach can only be detected while there is no downstream traffic on the bus. If power is removed from a port, it must respond by

registering a detach event and placing the port in the Powered Off state. This guarantees if a device is detached and a new one added that the attach event will be acknowledged.

The hub must ensure proper termination of a packet when a device disconnects during upstream signaling. This means that during upstream signaling the hub must limit the transmission of SE0 signal to less than 2.5 μsec, but not less than the longest valid EOP (which is about 16 FS bit times). In order to accomplish this, the hub may recognize an SE0 persisting for 23 FS bit times during upstream signaling as a "potential disconnect" event. Upon recognizing the “potential disconnect” event the hub must teardown upstream connectivity by driving its root port to the J state and then floating the bus. The upstream hub will therefore see a SE0 which is shorter than 2.5 μsec followed by a J state which will cause it to tear down upstream connectivity. The elongated se0 at the end of the packet will cause the packet to be rejected by the host as a packet with invalid EOP. Note that the hub still must time the SE0 signal for 2.5 μsec before considering a disconnect event to have occurred.

11.2.4 Bus State Evaluation

Whenever bus evaluation is performed two pieces of information are returned: whether a device is connected; and the speed of the device. Bus state evaluation is done at the end of the frame and is able to discriminate between the SE0, the differential 1 and 0 bus states.

Bus state evaluation should be done when no signaling is being driven on the bus. Since a port may be propagating packet traffic during most of the frame, this is most easily done at the end of a frame between the EOF1 and EOF2 points. Ports driving resume signaling should wait till the end of resume before evaluating bus status. Disconnected ports may evaluate bus state at any time since their outputs should never be driven.

Bus state evaluation can also be done after executing a PORT_RESET command. In this case, the hub should wait till the bus has floated to its quiescent state before evaluating status (the quiescent state is where one of the bus lines is Vih min for max load capacitance on the bus line). Therefore packet traffic through this port should be prevented during this float period . Packet traffic should be enabled such that no runt (short) packets are propagated downstream. Packet traffic should be enabled before the end of the frame so SOFs can propagate downstream in the next frame. If the (reset) device downstream is a hub, this hub should be able to use these SOFs for achieving frame lock.

Connect/Disconnect detect can only be performed after V_{bus} is applied to the downstream port. (This requirement only affects hubs whose downstream ports are power switched.) When a device is connected, the bus state changes from the disconnected to the attach detect state. Low speed devices pull up D- to an SE1 and leave D+ at SE0. Full speed devices pull up D+ to an SE1 and leave D- at SE0. Each downstream hub port must be capable of detecting and differentiating between low speed and full speed device connections once a device is connected. The differential J and K states are undefined until a device is attached and the device’s speed has been ascertained.

When a connect or disconnect occurs, it must be reflected in the hub status by the end of the frame in which the event occurred unless the hub is in the reset or suspend modes. A hub in the suspend mode is awakened by a connect or disconnect event and must be capable of reporting the event upon completion of resume. Upon coming out of reset, a hub must detect which downstream ports have devices connected to them. Connect and disconnect changes are reported on a per-port basis.

11.2.5 Full vs. Low Speed Behavior

Hubs must differentiate between full speed and low speed devices when a device is connected to the bus or at power-up. Hubs detect whether a device is full or low speed when the hub port transitions from its disconnected to its disabled state. Devices attached to a hub are determined to be either full speed or low speed by detecting which data line (D- or D+) is pulled high. Low speed devices pull D-

high, and full speed devices pull D+ high. Full speed signaling must not be transmitted to low speed devices. Doing so could cause low speed devices to mistakenly respond to full speed signaling and create a bus conflict. Communication between the host and the hub controller are always done using full speed signaling.

If a device is detected to be low speed, the hub port's output buffers are configured to operate at the slow slew rate (75-300 ns), and the port will not propagate downstream directed traffic unless it is prefaced with a preamble PID. Low speed signaling immediately follows the PID and is propagated to both low and full speed devices; but the sense of the differential data is inverted before it is sent down the LS port. Full speed devices will never misinterpret low speed traffic because no low speed data pattern can generate a valid full speed PID. When low speed signaling is enabled, a hub continues to propagate downstream signaling to all enabled ports until a downstream EOP is detected, at which time the output drivers for the low speed ports are turned off and will not be turned on again until the hub receives another PRE PID. If a port is disabled, no signaling is propagated to the port. Hubs must enable their low speed port drivers within four full speed bit times of having received the last bit of the PRE PID, and at that time they must drive a low speed J onto the bus.

If a downstream port is enabled, it propagates upstream directed bus signaling independently of whether the port was configured as low speed or full speed. Hubs implement slew rate selectable output buffers only in the downstream direction on their downstream ports; in the upstream direction, they transparently propagate both low and high speed traffic using fast (4-20 ns) edge rates. Low speed devices do not append a preamble onto their upstream traffic; however the hub will invert the sense of the LS differential signaling before propagating it to a FS port .While propagating low speed traffic upstream, hubs must be able to respond to EOPs that are either two low speed bit times or two full speed bit times wide. The former will occur under normal operation in which the low speed device generates the EOP. The latter occurs if a downstream hub encounters an end of frame babble or LOA condition and generates an EOP upstream in response (refer to Section 11.4.5).

11.2.5.1 Low Speed Keep-Alive

All hub ports to which low speed devices are connected must generate a low speed keep-alive strobe, generated at the beginning of the frame, which consists of two low speed bit times of SE0 followed by at least 0.5 bit times of J state. Low speed devices use the strobe to prevent themselves from going into suspend in the absence of low speed bus traffic. The hub repeater generates the keep-alive from its internal SOF counter, and the keep-alive strobe must start no sooner than the second EOF point and the SE0 must complete no later than the EOP of the token packet.

The low speed keep alive strobe must be generated once per frame, and it must track the SOF token such that the following rules are adhered to.

1. When going into suspend, the keep-alive must not go away before the last SOF.
2. A hub is allowed to synthesize no more than three keep alive strobes after receipt of the last SOF.
3. Keep alive must occur no later than one frame after resumption of SOF.

Keep-alive strobe generation that satisfies these requirements can be achieved by generating it from the decode of the PID of the SOF token; if the SOF token is lost or the PID is corrupted, the locked frame timer can be used to locate the keep-alive strobe generation.

11.2.6 Hub State Operation

The hub repeater needs to provide the following functions for packet traffic:

1. set up connectivity and tear down connectivity on packet boundaries.

2. prevent upstream connectivity when frame timer is not locked or when a downstream port is babbling.

An example implementation of a state machine to provide such functionality is shown in Figure 11-6. In normal operation, a hub idles in the WFSOP state, waiting for a start of a packet (SOP) to be detected on its root port or any of its enabled downstream ports. If an SOP is detected, the hub establishes connectivity originating from the port on which the SOP occurred and transitions to its WFEOP state. It remains in this state until an end of packet (EOP) is encountered or until the end of frame (EOF) occurs. Under normal circumstances and when not near the end of a frame, a hub repeater will transition back and forth between WFSOP and WFEOP.

A hub will transition to the WFDSOP state upon coming out of resume or upon reset (either power-on reset or reset at the root port). Doing so guarantees that traffic cannot propagate upstream (and possibly lock up the bus) until the hub's frame timer has achieved lock with the host. Details of the interaction between the frame timer and the hub repeater state machine are detailed in Section 11.5.1.1.

A hub in the idle (WFSOP) state responds to the end of frame (EOF1) point by transitioning to the WFDSOP state. Transitions from WFSOP and WFEOP to WFDSOP are not errors, but simply indicate that the hub is nearing the end of its frame and cannot establish connectivity until the start of the next frame.

WFEOF2 is a special state which is entered only when a babble or loss of bus activity (LOA) is detected near the end of a frame and upstream connectivity is established. If a hub repeater is still in the WFEOP state (i.e., it has not received an EOP) when the EOF1 point is encountered, it transitions to the WFEOF2 state. It will remain there until its EOF2 point or an upstream EOP occurs, at which time it transitions to WFDSOP and awaits the next Downstream SOP (DSOP), which will normally be the SOP associated with the SOF packet, and indicates the start of the next frame.

When a DSOP occurs, the hub returns to the WFEOP state and waits for the end of the packet. If EOF1 occurs while the hub still is maintaining downstream connectivity, the hub will maintain connectivity but transition to the WFDSOP state because no upstream connectivity is allowed to be established after the EOF1 point and until after the next downstream (SOF) packet.

If a hub is still in its WFEOF2 state when EOF2 occurs, the port that established upstream connectivity must be disabled, regardless of the bus state. If EOF2 occurs and the hub is in the WFDSOP state but the bus at an enabled port is not in the idle state, that port must be disabled since such activity indicates a babble condition.

Note that upstream connectivity is terminated if a downstream port is seen to be babbling at the end of frame or if a disconnect is detected in the middle of an upstream packet. This is shown in the state diagram with the UPST_ERR transition from WFEOP to WFDSOP.

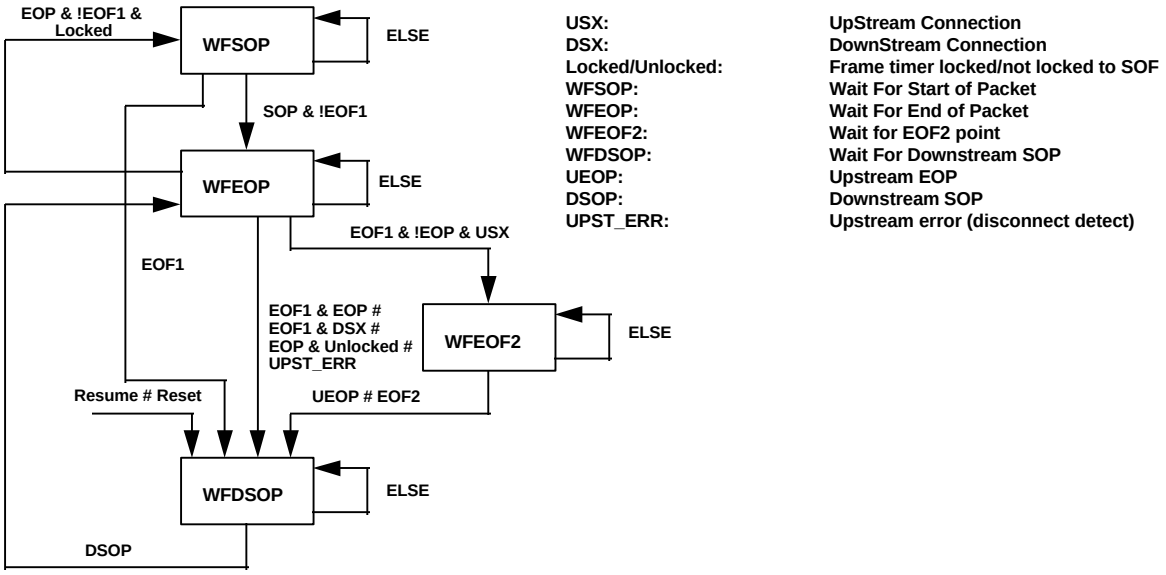

Figure 11-6. Hub Repeater States

The hub repeater maintains state across each packet that is detected and repeated by the hub. The repeater state machine does not need to track more than a single packet and need not, for example, track across multiple packets in a transaction. Figure 11-7 shows how hub states change in the course of a normal packet transmission.

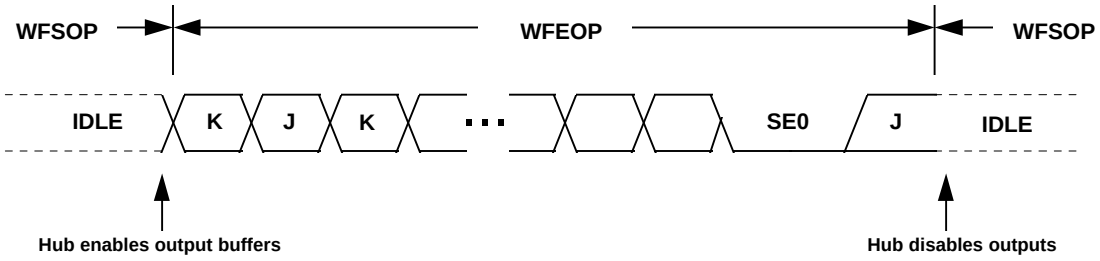

Figure 11-7. Hub States Across a Packet

11.2.6.1 Hub Repeater States

For upstream connections, a hub repeater may transition between four states: wait for start of packet (WFSOP), wait for end of packet (WFEOP), wait for EOF2 point (WFEOF2), and wait for downstream start of packet (WFDSOP). The EOF1 and EOF2 points are described in Section 11.4.5.1. The four states are described below. For downstream connections, the hub may transition between WFSOP, WFEOP, and WFDSOP states.

11.2.6.1.1 Wait for Start of Packet

The Wait for Start of Packet (WFSOP) is the state a hub occupies when there is no packet currently being propagated to or through the hub and the hub has achieved the frame lock condition with the host. In the WFSOP state, all of the hub's enabled ports are in the receive mode with their output buffers in the Hi-Z state. If the root port or any enabled downstream port detects an SOP, the hub establishes connectivity and transitions to the Wait for End of Packet state.

11.2.6.1.2 Wait for End of Packet

During the Wait for End of Packet (WFEOP) state, the hub has established its connectivity and is receiving packet traffic on one of its ports. The hub transparently propagates the traffic in either the upstream or downstream direction. A hub transitions out of the WFEOP state when it detects an EOP or if it encounters an end of frame (EOF1) point (refer to Section 11.4.4). Detection of EOP causes the hub to transition back to WFSOP and is the normal sequence when frame lock exists. If frame lock does not exist then the detection of an EOP causes a transition to the WFDSOP state. If EOF1 is detected and an upstream connection exists, the hub transitions to the WFEOF2 state. If the EOF1 occurs during a downstream connection, the hub will transition to the WFDSOP state and wait for the next downstream SOP before allowing any upstream connectivity.

11.2.6.1.3 Wait for EOF2 Point

The WFEOF2 state is entered only when the hub detects its EOF1 point and is still waiting for an EOP from a downstream port. This condition is potentially indicative of babble or loss of bus activity. A hub repeater remains in the WFEOF2 state until an EOP is detected or until its EOF2 point occurs.

11.2.6.1.4 Wait for Downstream SOP

A hub repeater enters the Wait for Downstream Start of Packet(WFDSOP) state either when EOF1 is detected and the hub is in the WFSOP state (normal end of frame behavior); or when an UPST_ERR is seen in WFEOP state (potential disconnect behavior - see section 11.2.3.1) or when the hub is in the WFEOF2 state and an EOP or EOF2 point is detected (babble/LOA) behavior. The hub repeater also enters this state upon hub resume or upon reset because frame lock must be achieved before allowing upstream connectivity.

11.3 Hub I/O Buffer Requirements

All hub ports must be able to detect and generate the J, K, and SE0 bus signaling states. This requires that hub ports be able to independently drive and monitor their D+ and D- outputs. Each hub port must have single ended receivers on the D+ and D- lines as well as a differential receiver. Details on voltage levels and drive requirements appear in Chapter 7. Figure 11-8 shows an example of the I/O circuitry for a typical hub port.

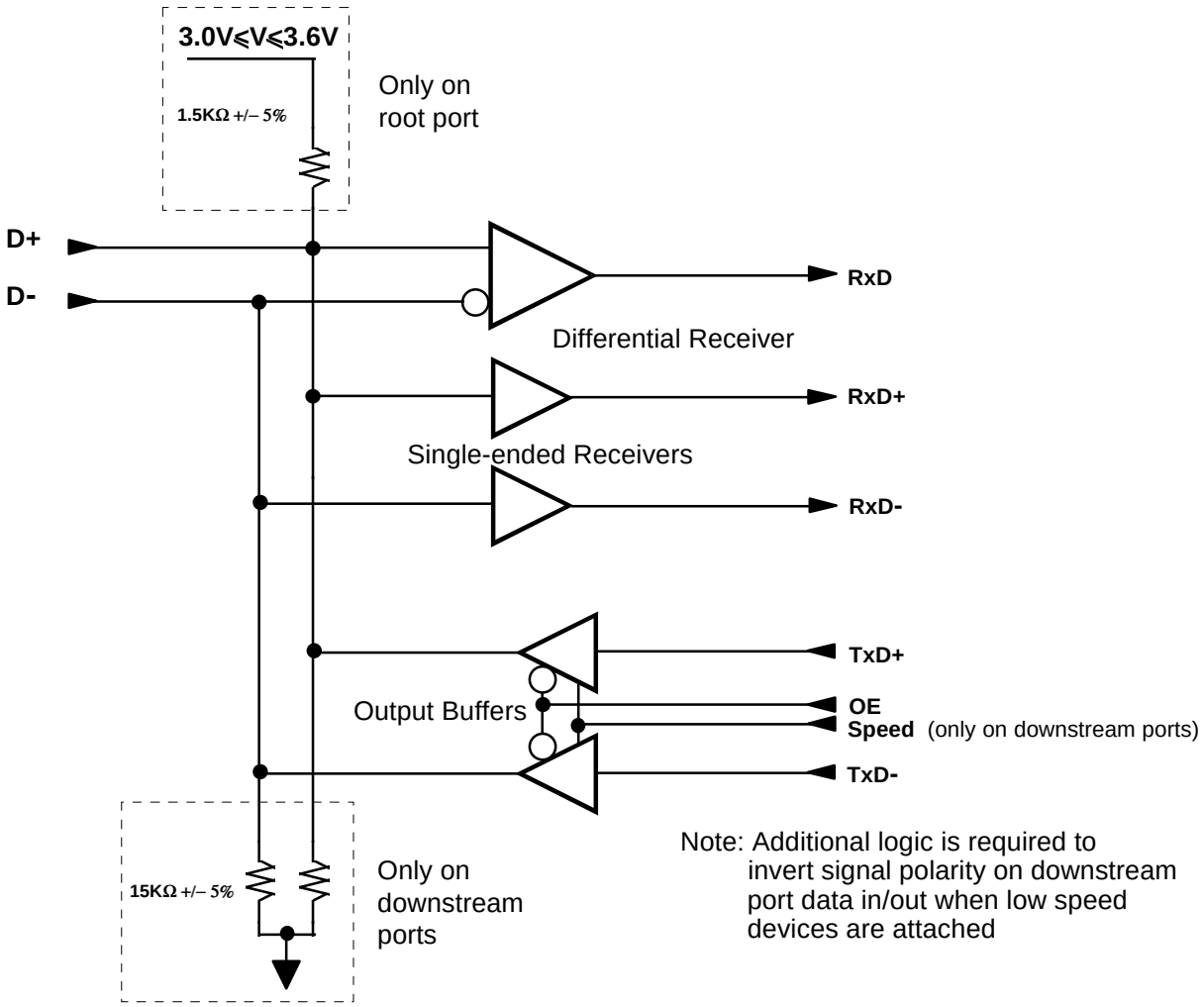

Figure 11-8. An example of a Hub Port I/O Driver and Receiver

Table 11-2 defines the hub I/O section's input and output signals.

Table 11-2. Hub I/O Section Signals

Signal Name	Direction	Description
D+, D-	I/O	External USB data lines
RxD	O	Received differential data
RxD+	O	Received single-ended value on D+ line
RxD-	O	Received single-ended value on D- line
TxD+	I	Transmitted data value
TxD-	I	Transmitted data value
OE	I	Output enable/disable on output buffers

D+ and D- are the I/O lines that connect to the USB physical medium. When placed in the Hi-Z state, they are pulled to near the ground or Vcc rails by resistors on the hub and device. RxD is the differential received data. RxD+ and RxD- are the received single ended data. TxD+ and TxD- are used to send differential data and single ended reset and EOP signaling. OE disables the output drivers.

11.3.1 Pull-up and Pull-down Resistors

Hubs, and the devices to which they connect, use a combination of pull-up and pull-down resistors to control D+ and D- in the absence of their being actively driven. These resistors establish voltage levels used to signal connect and disconnect and also maintain the data lines at their idle values when not being actively driven. Each hub downstream port requires a pull down (R_{pd}) on each data line; the hub root port requires a pull-up (R_{pu}) on its D+ line. Values for R_{pu} and R_{pd} appear in Chapter 7.

11.3.2 Edge Rate Control

Downstream hub ports must support both low speed and full speed edge rates. Full speed signaling specifies a rise/fall time of 4-20 ns. Low speed rise/fall times must be within a 75-300 ns range. Edge rate on a downstream port must be selectable, based upon whether a downstream device was detected as being full speed or low speed. The hub root port always uses full speed signaling, and its output buffers always operate with full speed edge rates.

11.4 Hub Fault Recovery Mechanisms

Hubs are the key USB component for establishing connectivity between the host and other devices. It is vital that any connectivity faults, especially those that might result in a deadlock, be detected and prevented from occurring. Hubs need to handle connectivity faults only when they are in the repeater mode. Hubs must also be able to detect and recover from lost or corrupted packets which are addressed to the hub controller. Since the hub controller is, in fact, another USB device, it must adhere to the same time-out rules as other USB devices, as described in Chapter 8.

11.4.1 Hub Controller Fault Recovery

The hub controller must be able to respond to and recover from corrupted and missing packet transmissions. These include lost or corrupted token, data, and handshake packets. The following table describes the possible field level errors which the hub controller can detect and its responses.

Table 11-3. Packet Error Types

Field	Error	Action
PID	PID check, bit stuff	Ignore packet
Address	Bit stuff, address CRC	Ignore token
Data	Bit stuff, data CRC	Discard data

11.4.2 False EOP

Hub handling of false EOP differs depending on whether the hub is operating as a repeater or is being accessed as a device. A hub operating as a repeater transparently propagates signaling, and cannot differentiate between a “good” EOP and a “false” EOP. If any EOP occurs, the hub tears down connectivity and waits for the next SOP. If the packet transmitter continues sending, the hub re-establishes connectivity on the next J to K transition. From a hub’s point of view, a false EOP makes a single packet look like two separate packets. The hub does not participate in false EOP error detection or recovery process when operating in the repeater mode.

When a hub is accessed as an USB device, the hub controller detects and recovers from false EOP the same as any other USB device, as described in Section 8.7.3.

11.4.3 Repeater Fault Recovery

Hubs must be able to detect and recover from conditions that leave them waiting indefinitely for an end of packet or that leaves the bus in a state other than the idle state at the end of a frame, i.e., an SOP without an EOP. There are two such hub fault conditions: loss of activity and babble. Loss of activity (LOA) is characterized by detection of a start of packet (SOP) followed by lack of bus activity (bus is stuck at the idle or K state) and no end of packet (EOP) by frame’s end. Babble is characterized by SOP followed by the presence of bus activity continuing past the end of a frame. Hubs have no notion of allocated bandwidth and must rely upon the frame timer to detect LOA and babble conditions. The recovery mechanism requires that hubs track the host’s frame timing and recover before the beginning of the next frame.

Hub fault recovery operates only in the upstream direction. The host is responsible for detecting and recovering from its own downstream directed errors. Hub repeater fault recovery must meet the following requirements:

- Devices driving illegal states at the end of a frame must be isolated from the bus by disabling the downstream hub port to which the device is connected.
- Hubs must return the bus to the idle state before the start of the next frame if the connectivity was previously established in an upstream direction.

Under non-fault conditions, these requirements are met by virtue of a hub receiving an EOP with every packet and having no bus traffic occur past the end of a frame. Before describing how hubs implement fault recovery, the hub frame timer will be described.

11.4.4 Hub Frame Timer

Each hub has a frame timer whose timing is derived from the host-generated SOF token and tracks the host SOF packet in both phase and period. The frame timer is reset each time an SOF is detected and is responsible for generating End of Frame (EOF) points. The hub frame timer must track the host SOF and be capable of remaining locked to the host SOF for the loss of up to two consecutive SOF tokens. All hubs must have an EOF timer, and it is used to identify two distinct points in time: a point (EOF1) at which time the hub must terminate upstream connectivity, and a point (EOF2) by which time the bus must see upstream traffic terminated. The delay between EOF1 and EOF2 corresponds to the timing skews between the host and the hub plus time required for EOP to occur and is illustrated in Figure 11-9.

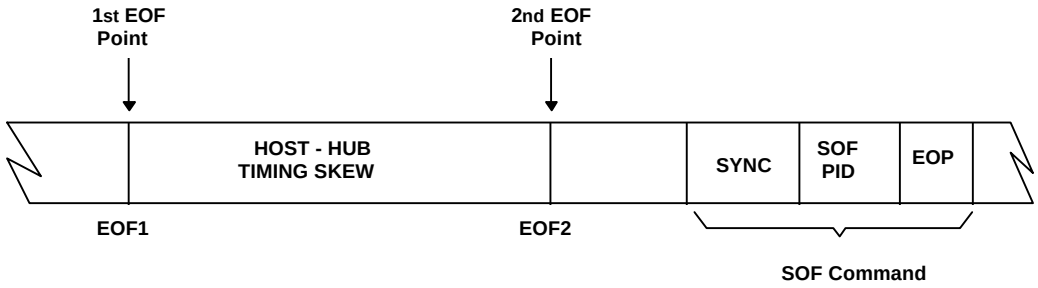

Figure 11-9. Host-Hub EOF Skew

The frame timer must lock to the host's frame timing for worst case tolerances and offsets between the host and hub. The offsets have to accommodate the hub oscillator tolerance and purposely introduced deviations from 1.000 ms frame timing which occur when the host locks to an external source. The above parameters require a minimum frame length (FL_{min}) of 11,955 and a maximum frame length (FL_{max}) of 12,045. The +/-45 count is the cumulation of +/-30 due to the +/-0.25% data rate tolerance, and +/-6 due to the +/-0.05% frame interval accuracy and +/-9 due to possible host-hub skew as described in section 11.4.5.2.

11.4.4.1 Frame Timer Locking

The frame timer is clocked from the hub's clock source and is locked via SOF packets to the host's frame time. When a hub is first connected to the host the hub's frame timer is not locked to the host. When the first SOF token is detected the frame timer resets and starts counting once per 12 MHz clock cycle; it continues to count up until the next SOF token is detected. At this time the frame timer is reset and the previous count value is stored as the previous frame length (PFL) count. This value is updated for each frame in which an SOF is detected and represents, in hub clock counts, the host's frame time. The frame timer is considered locked to the host when the difference between the PFL count and the current count at the end of a frame is less than or equal to 2 bit times and no SOFs are lost.

Should an SOF token fail to be received when locked, the hub uses the previous frame length count as a best approximation to the current frame length. The hub will do this for the loss of up to two consecutive SOFs i.e. a locked frame timer is defined as being locked to the host for the loss of up to two consecutive SOF tokens. Hubs must be able to lock to the host within 3 frames after detecting the first SOF packet (assuming no missing SOF packets). When a hub comes out of resume its frame timer is not locked with that of the host. During this time, when the hub is in the awake state, it must be prevented from establishing upstream connectivity until the hub frame timer is locked to the host.

The hub's error recovery, as described in Section 11.4.5, is operational whenever the hub frame timer is locked.

11.4.5 Hub Behavior Near EOF

Hub behavior near the end of frame is diagrammed in Figure 11-10. There are two end of frame timing markers, EOF1 and EOF2, corresponding to the first and second EOF points. All hubs receiving upstream traffic, upon detection of EOF1, transmit a valid EOP upstream (as defined in Section 7.1.11.2), and then float the bus. Starting with EOF1, hubs are not permitted to re-establish upstream connectivity until the end of the next downstream packet which is usually the next SOF token. The hub's state controller only evaluates EOF1 and EOF2 when the hub's frame timer is locked. If a hub's frame timer is not locked the hub is not permitted to establish upstream connectivity.

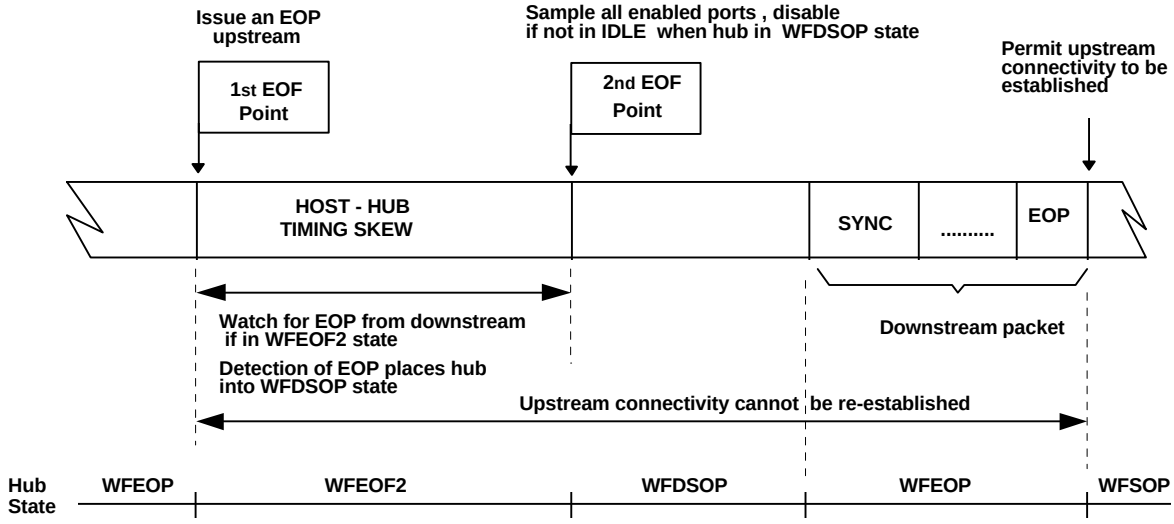

Figure 11-10. Hub Behavior Near EOF; (upstream connectivity and hub in WFEOP at EOF1)

1. Hub behavior at the end of frame is summarized below (states mentioned refer to the example implementation described in Figure 11-6). Hub recovery behavior is applicable only for upstream packet traffic (i.e. at enabled ports). If a hub encounters the EOF1 and downstream connectivity is established, it transitions to the WFDSOP state but maintains connectivity for the remainder of that packet. This is so that no upstream connectivity would be established in the time from the EOF1 point and until after the next downstream packet.
2. At their EOF1 point, all hubs with established upstream packet connectivity transmit a valid upstream EOP, and then float the busThe EOP must be of sufficient width to cause the upstream hub to tear down connectivity but not so wide as to be misrecognized as a disconnect.
3. Hubs will not allow further connectivity to be established in the upstream direction after EOF1.
4. Hubs that have established upstream connectivity and haven't seen EOP by EOF1 must watch for EOPs on the downstream port on which connectivity was established. They must monitor the bus from EOF1 to EOF2 (example state machine is in WFEOF2 during this period) .
5. If an EOP from downstream is detected by a hub in the WFEOF2 state in the EOF1 to EOF2 window, the hub will transition to the WFDSOP state.
6. At the EOF2 point enabled ports are disabled if IDLE is not present on that port or if upstream connectivity is established.

11.4.5.1 Skew Requirements

Hub frame timers, while all locked to the host's frame timer , are subject to certain skews which dictate the length of time between the EOF points, host behavior near EOF, and the next SOF. Figure 11-11 illustrates critical end of frame timing points.

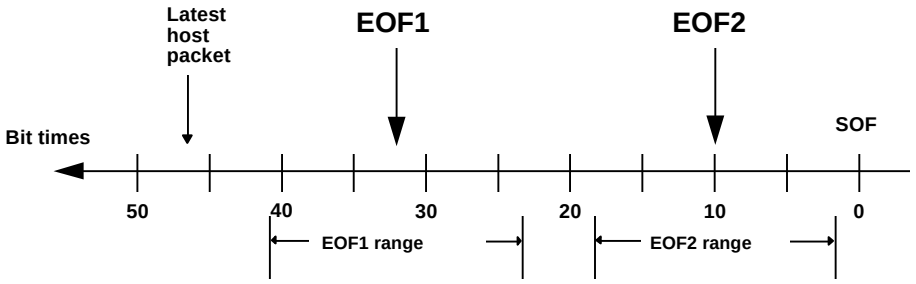

Figure 11-11. EOF Timing Points

11.4.5.2 Host-Hub Skew

The timing skew between the host's SOF point and the hub's SOF timer is minimized by the requirement that the hub track the host. Sources of skew include the fact that hubs may miss SOFs, and that the host frame counter can be adjusted to track an external master clock. The 12 MHz clock is the only clock actually specified, so this is the best granularity available by specification. Assuming a fixed host SOF timing and that two consecutive SOFs can be missed, the maximum cumulative host-hub skew without host timing wander is ±3 clocks. Assuming that the host clock may be adjusted by up to one bit time per frame, then the host can walk away from the hub by 1 + 2 + 3 = 6 clocks. The maximum host-hub skew is the sum of these two components or ±9 clocks.

The second EOF point must be sufficiently separated from the SOF point to permit hubs to recover and be ready to receive the SOF token from the host. A hub must finish sending its EOP before a hub to which it is attached reaches its second EOF point. This means that all hub EOF2 points must occur at least one bit time before the host issues SOF. All hub EOF2 points must lie within a ±9 bit time window; therefore, EOP must lie outside this window and complete at least 2 x 9 + 1 = 19 bit times before host SOF.

The next step is calculating how long it takes to generate EOP and how far back from SOF it must occur. Transmitting EOP requires four bit times. Therefore, a hub must start sending its EOP no later than 19 + 4 = 23 bit times before SOF. For a hub to be sure that it starts no later than the 23rd bit time, it must start 9 bit times before that or at bit time 32, which is the value of the first EOF point. The earliest that a hub might start sending EOP is 9 bit times before the first EOF point or at bit time 41.

A hub must not see a packet from the host start after the hub reaches its first EOF point. This could be as early as 41 bit times before SOF. Hub propagation delay must also be figured into the delay budget. The per-hub delay is approximately one bit time; so for a worst case topology of six hubs away from the host, there will be an additional 6 bit times of delay. Therefore, the host's EOF point for transmit is 41 + 6 = 47 bit times from SOF, relative to the host's SOF timer. If the host is still transmitting at bit 47 and not able to complete before SOF, it must force an error via a bit stuffing violation (at least eight 1's), followed by an EOP. If the host is still receiving a packet or an EOP at bit 41, it should treat the packet as being in error. Table 11-4 summarizes hub and host EOF timing points.

Table 11-4. Hub and Host EOF Timing Points

Description	Number of Bits From Start of SOF	Notes
EOF1	32	End of frame point #1
EOF2	10	End of frame point #2
Host invalidates full speed Tx packet	47	Latest that host may start a full speed packet (handshake)
Host invalidates low speed Tx packet	184	Latest that host may start a low speed packet (handshake)(rounded up to the nearest low speed bit time)
Host invalidates Rx packet	41	Host treats any packet still being received at bit time 41 as bad

11.5 Suspend and Resume

Hubs must support suspend and resume both as a USB device and also in terms of propagating the suspend and resume signaling. Hubs support both global and selective suspend and resume. Selective suspend and resume are implemented via per port suspend/resume; selective resume can also be initiated by a downstream device. Global suspend is implemented by the host through the hub's root port. Global resume may be initiated either from the host or from a hub's downstream port.

11.5.1 Global Suspend and Resume

Global suspend is initiated by the host shutting off downstream traffic to the entire bus. A hub enters the suspend state if it detects a continuous idle state on its root port for at least 3.0 ms. When placed into the suspend state, a hub floats all of its output drivers, maintains static values of all its control and status bits, and preserves the current state information for each of its downstream ports.

Hub resume may be initiated by any bus transition on a hub's root port or by resume (K) signaling at a downstream port in the enabled or suspended state. When the DEVICE_REMOTE_WAKEUP feature is set, a hub resume may also be initiated by the connect/disconnect of a device on a downstream port. If an idle to a resume bus transition occurs on an enabled or suspended downstream port then the hub immediately reflects an idle to a resume bus transition to that port, all other enabled downstream ports, and its root port. While in global suspend the hub may initiate resume for hub originated events (e.g. connect/disconnect on downstream ports) if and only if the DEVICE_REMOTE_WAKEUP feature is set. Nonetheless, the state of DEVICE_REMOTE_WAKEUP does not prevent the hub from propagating resume signaling originating on either the hub's upstream or downstream ports. Figure 11-12 shows the timing relationships during a resume sequence in which a device initiates a wake up to a suspended hub.

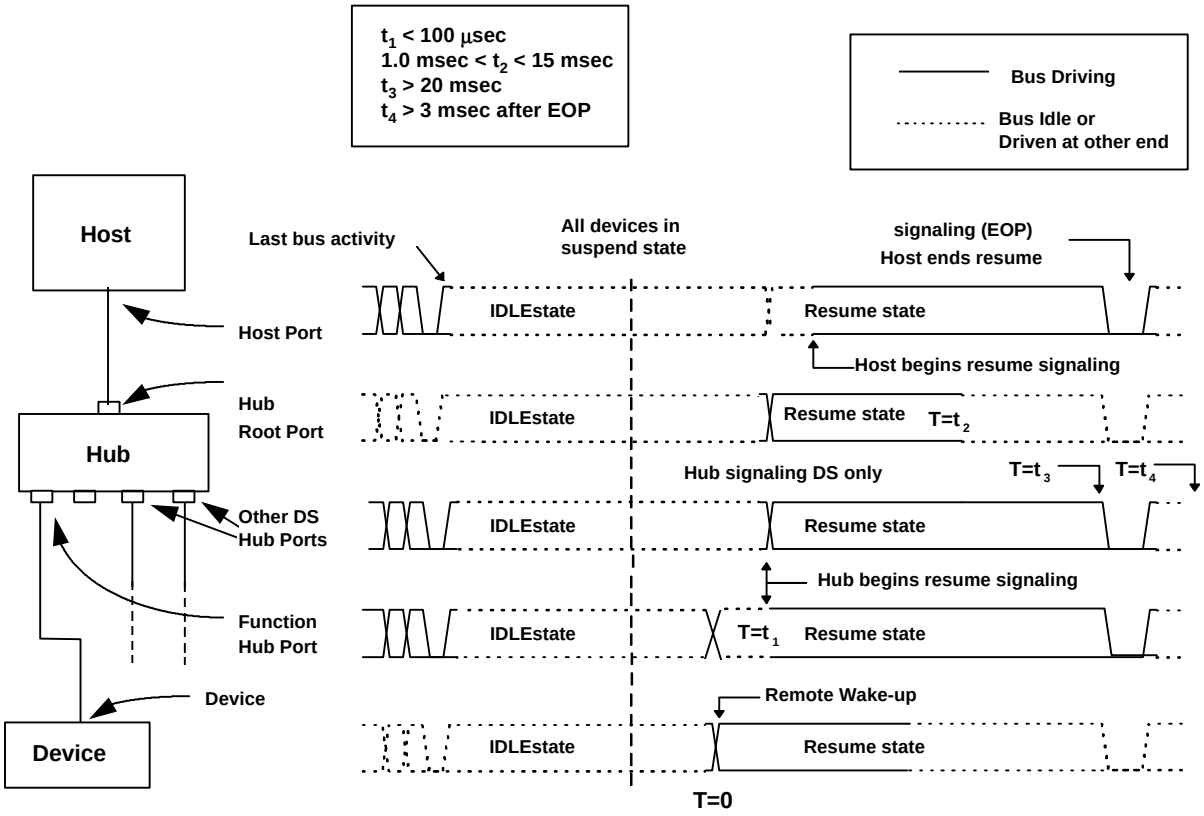

Figure 11-12. Resume Signaling

There are four time parameters that hubs must conform to, as shown in Figure 11-12. T=0 represents the time when the resume signaling arrives at the port. t_1 is the time by which a hub must respond by driving resume onto the upstream port, enabled downstream ports and the port that initiated the resume. t_2 is the time at which the hub must stop driving an upstream initiated resume to its root port and reflect the bus state now being driven to its root port onto its downstream ports. The interval, t_3 is the time that a hub must generate downstream K signaling to the device that initiated the resume. t_4 is the time that the hub must wait after detecting the downstream low speed EOP until it can set its interrupt bit, indicating that a resume has occurred after a selective suspend.

When a device drives an idle to resume transition upstream to the hub the hub responds by driving a resume state onto the bus to the root port and to all enabled downstream ports including the port which initiated the resume within 100 us of the resume event. (It is acceptable to drive both ends of a wire to the same state). The hub is required to drive this K state for at least 1 ms. After 1 ms the hub is guaranteed that the upstream hub (or host) will be driving a K state downstream. So at t_2 the hub can set up downstream connectivity and use this K state from upstream to sustain the K state that it has been driving downstream. The hub must have set up this downstream connectivity before 15 ms from the start of the resume.

The resume signal propagates upstream through other hubs in the path until it reaches an awake hub or the host (which is always awake) . The port at the awake hub reflects the resume signaling downstream for at least 20 ms, which guarantees that all downstream hubs and devices on enabled paths will have time to wake up and detect the downstream resume signaling. The host terminates the resume sequence by driving an EOP for two low speed bit times. The EOP is interpreted as a valid end of packet, causes all hubs to tear down their connectivity and informs all devices on the bus that the resume sequence has completed. The device that initiated the resume must wait until it detects EOP to determine that the resume sequence has completed.

Hubs must be able to propagate downstream traffic immediately after the end of resume to prevent downstream devices from re-entering the suspend state. The hub controller must be able to receive packet traffic no later than 10 ms after the end of resume. Note: a remote wakeup device may not start a resume sequence until 5 ms after the last bus activity. This allows the hub to go into the suspend state so that it will resume all ports and not just the full speed ones.

Suspended hubs can also be woken up by signaling from the host (global or directed resume). Table 11-5 summarizes the behavior of suspended hubs in response to host initiated, downstream signaling. Note that connectivity is established through enabled ports even though the hub is suspended. This allows all devices on enabled paths to be woken up by a single global resume sequence.

Table 11-5. Suspended Hub Behavior During Global Resume

Downstream Port Status, Signaling at root port	Downstream Port Response
Port enabled, resume (K) received	Signal resume downstream
Port disabled, resume (K) received	Do nothing
Port suspended, resume (K) received	Do nothing

11.5.1.1 Resume and Hub Frame Timer

When a hub transitions from the suspended to the awake state, its frame timer is not operational and the hub's EOF timing recovery circuitry will not work until the frame timer has locked to the host by using the SOFs. To prevent a babbling downstream device on a recently resumed bus segment from locking up the bus it is necessary to prevent the hub from propagating upstream traffic while the hub is awake, but its frame timer has not yet locked to SOF timing from the host. This is achieved by making the hub repeater state machine transition to its WFDSOP state upon coming out of resume. While the hub repeater is in the WFDSOP state all upstream traffic is ignored. The repeater state machine remains in the WFDSOP state until a DSOP is detected, at which time it transitions to the WFEOP state. Downstream traffic is unaffected by the status of the frame timer. To prevent needless time-outs it is recommended that the host not send any packets addressed to devices on a just resumed bus segment until the host has issued at least two SOF tokens. This condition is guaranteed by virtue of the fact that a hub does not report a resume complete to the host until 3.0 ms after the resume sequence completes.

11.5.2 Selective Suspend and Resume

Selective suspend and resume provide a means for placing a single device or a bus segment into a low power state. Selective suspend relies on the ability of a hub to selectively suspend individual ports via a SetPortFeature(PORT_SUSPEND) request, which places a port on a hub (referred to as the disabling hub) into the suspend state (see Figure 11-4 and Figure 11-5 for state diagrams). In the suspend state a port is prevented from propagating any bus activity (except the port reset or port resume request) downstream, and the port can only reflect upstream bus state changes via the hub's status bits; i.e., an awake hub cannot propagate upstream traffic from a suspended port to its root port. The hub must also insure that the port accessed via the port suspend request is not suspended in the middle of a packet. In response to a port suspend, all devices downstream of the targeted port go into suspend after failing to see bus activity for 3.0 ms while the bus is in the idle state. The port suspend request is only understood by an awake hub. If the host wishes to send the request to a suspended hub, it must first wake the hub and then issue the desired request.

11.5.2.1 Selective Resume to an Awake Hub

Figure 11-14 shows the traffic and resume signaling connectivity for an awake hub when it receives remote wakeup, connect, and disconnect events. An awake hub must prevent resume signaling from propagating to any port except the port on which the resume occurred. A remote resume sets the corresponding status and status change bit in the hub controller.

11-13The receipt of resume (K) signaling by Port B of Hub Y causes the following sequence of events to occur.

1. Hub Y, Port B reflects the resume signaling downstream within 100 µs of resume detection.
2. Hub Y maintains resume signaling downstream for at least 20 ms.
3. Hub Y terminates the resume with a low speed EOP and transitions port B to the enabled state (Downstream traffic starts flowing through the port at this time.)
4. 3.0 ms after the end of EOP, an interrupt in the hub controller is set.

A disconnect event on an enabled, disabled, or suspended port does not propagate to any other of the hub's ports, nor does it cause resume to be reflected to the originating port. A disconnect does set the disconnect status and status change bits in the hub controller. Similarly, a connect event to a disconnected port does not propagate or reflect to any port, but sets the connect status and status change bits.

Downstream selective resume to a port may also be initiated via the ClearPortFeature(PORT_SUSPEND) request. This request causes the hub controller to drive resume signaling onto the port for at least 20 ms, followed by a low speed EOP. At the end of the resume sequence the hub port's status and status change bits are set to indicate the resume request is complete. Note that the status bits may not change until 3.0 ms after the EOP is issued. Table 11-6 summarizes the behavior of ports in different states on an awake hub in the presence of downstream resume events.

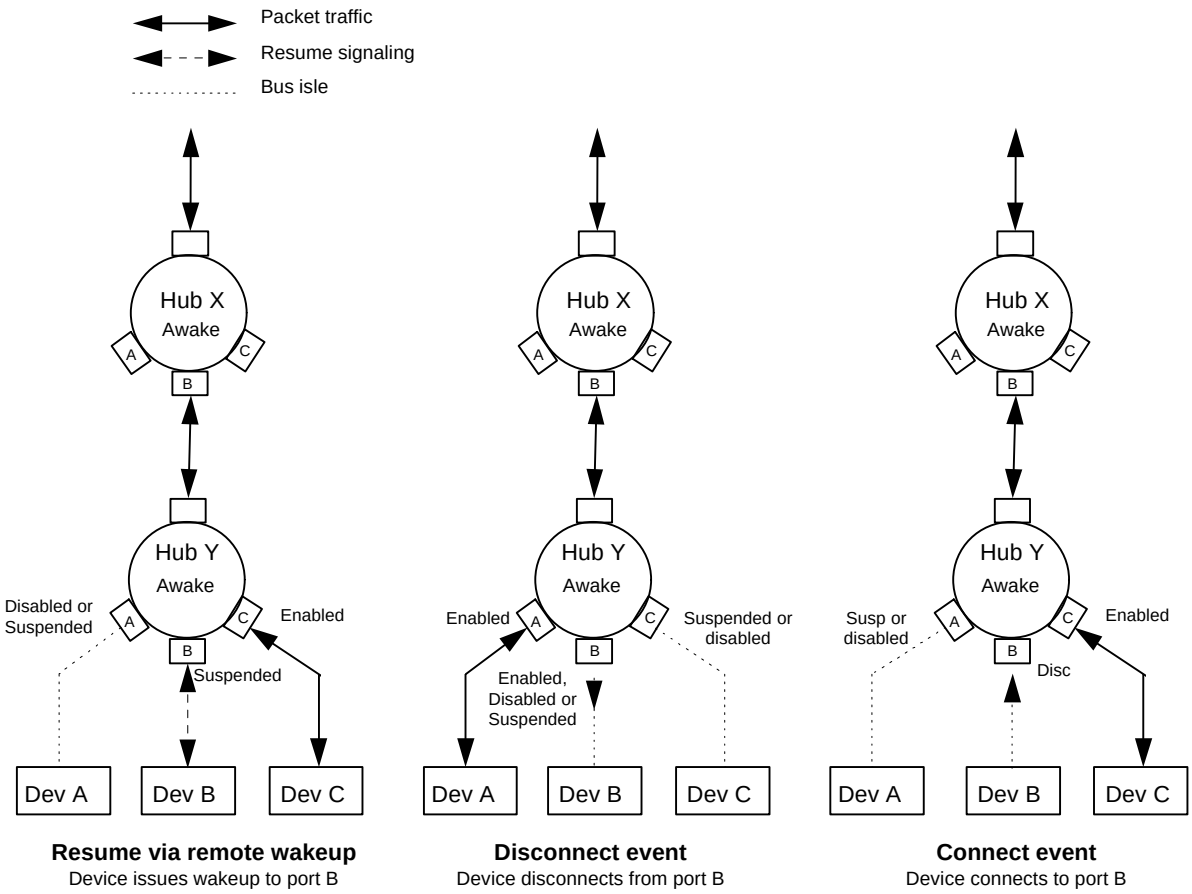

Figure 11-14: Signaling to an Awake Hub

Table 11-6. Awake Hub Behavior During Resume

Downstream Port Status and Signaling Type	Signaled Port Response	Adjacent Enabled Port Response	Adjacent Disabled Port Response	Adjacent Suspended Port Response
Port suspended, resume (K) received	Reflect K downstream on signaled port. Initiate port wake-up. Set status bits and interrupt.	Do nothing	Do nothing	Do nothing
Port enabled, disabled or suspended and disconnect received	Set port disconnect, and change status bits. Set interrupt.	Do nothing	Do nothing	Do nothing
Port disconnected and connect received	Set port connect, and change status bits. Set interrupt.	Do nothing	Do nothing	Do nothing

11.5.2.2 Selective Resume to a Suspended Hub

It is possible for the host to suspend a hub port and then suspend the entire hub. In this case, the hub's connectivity changes from packet signaling connectivity to resume signaling connectivity when the hub enters a suspended state. A remote wakeup event can occur at this hub in one of the five scenarios listed in the first column of Table 11-7. (More complicated scenarios are covered in the next section). The signaling at the various hub ports in response to this event are detailed in the other columns of this table. Figure 11-15 shows this in a pictorial manner. The signaling at port B of hub X has already been discussed as one of the scenarios in the previous section 11.5.2.1. Disconnect and connect events which generate upstream signaling will be inhibited from doing so if the hub is not programmed as a remote wakeup device as described in sec. 11.5.1. The figure depicts the simplest cases of remote wakeup. The wakeup timing for the hub and device have been described earlier in sec. 11.5.1. Simultaneous wakeup events and pending events are discussed in sec. 11.5.3.

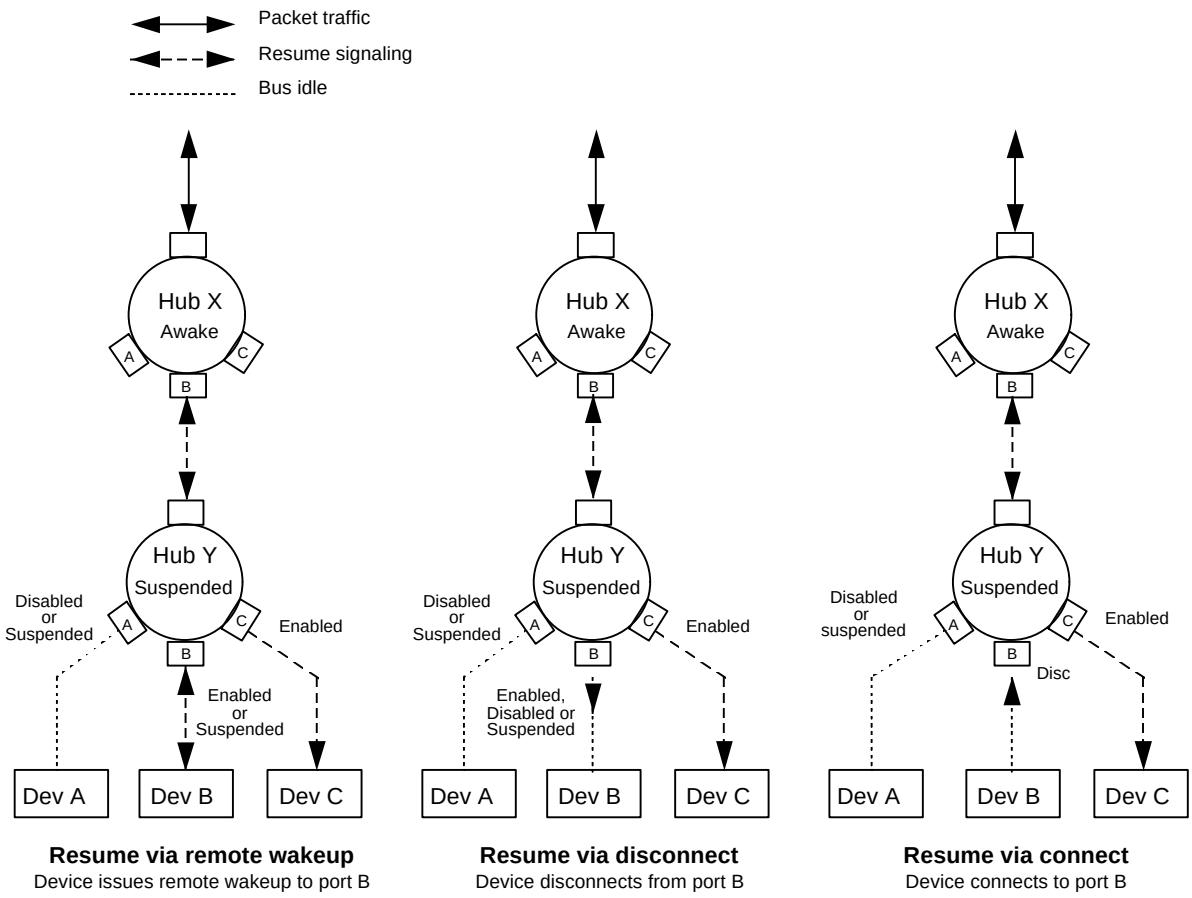

Figure 11-15. Signaling to a Suspended Hub

Table 11-7. Suspended Hub Behavior During Resume

Downstream Port Status and Signaling Type	Signaled Port and Root port Response	Adjacent Enabled Port Response	Adjacent Disabled Port Response	Adjacent Suspended Port Response
Port enabled, resume (K) received	Reflect resume upstream and downstream. Do not set status bits	Signal resume downstream	Do nothing	Do nothing
Port disabled, resume (K) received	Do nothing	Do nothing	Do nothing	Do nothing
Port suspended, resume (K) received	Reflect K upstream on root port and downstream on signaled port. Set suspend change status bit, set interrupt.	Signal resume downstream	Do nothing	Do nothing
Port enabled, disabled or suspended and disconnect received	Reflect resume upstream. Update port connect and change status bits, set interrupt	Signal resume downstream	Do nothing	Do nothing
Port disconnected and connect received	Reflect resume upstream. Set port connect, and change status bits, set interrupt	Signal resume downstream	Do nothing	Do nothing

Host initiated signaling for suspended hubs is shown in Figure 11-16. Waking up a device at the bottom of a string of suspended hubs is a multi step procedure which is described below.

The host takes Port B in Hub X out of suspend by issuing a port resume request to the hub. In response to the resume request, Hub X initiates the resume signaling by driving at least 20 ms of K signaling followed by a low speed EOP out its Port B. Hub Y sees the resume signaling and starts its wake up process. The EOP indicates that the resume sequence is completed and Hub X, port B is in the enabled state and Hub Y is awake.3.0 ms after the EOP is issued by Hub X, the resume complete status bit in Hub X's controller is set indicating that the resume sequence is completed.

The next step is taking Port B on Hub Y out of the suspended state to the enabled state. The procedure is identical to that described in the previous step except that the request is issued to Hub Y, Port B instead of Hub X. At the end of this step, Port B on Hub Y is enabled and device B has received resume signaling and is awake. At this time the host may communicate with Device B. This resume policy permits the bus to be sequentially suspended while permitting any device on the suspended segment to awaken the bus. It also permits nested hubs to be awakened one tier at a time.

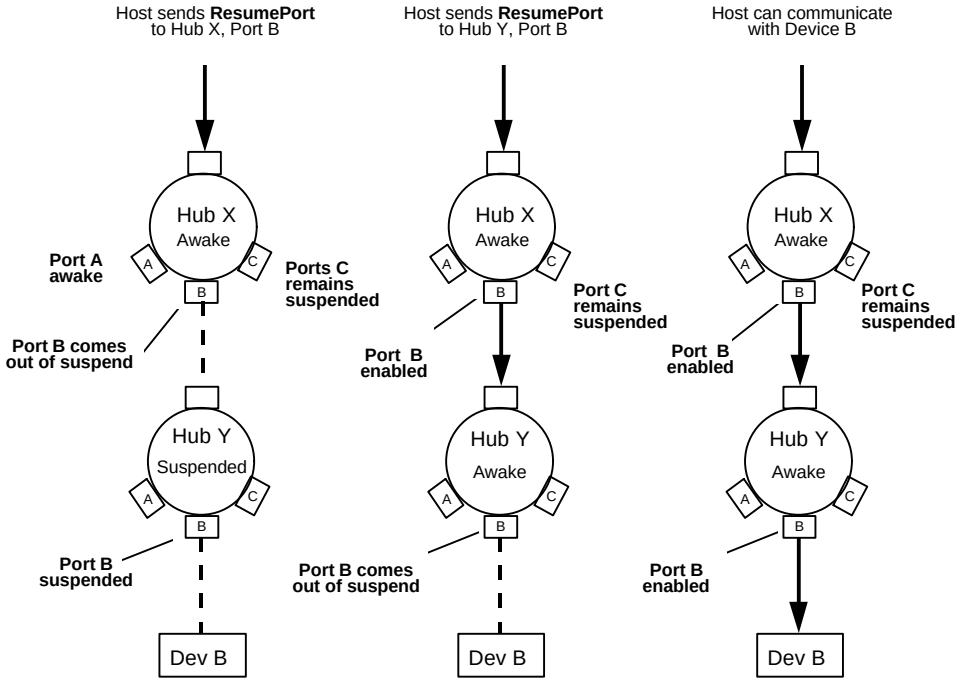

Figure 11-16. Host-initiated Resume Through Suspended Hubs

A J-->K event at the root port of a suspended hub causes the hub to establish downstream connectivity to all enabled downstream ports. This causes the resume signaling to propagate downstream and wake up all hubs and devices in downstream enabled paths.

A J-->se0 event at the root port of a suspended hub causes the hub to wake up. After the signaling is determined to be a valid reset, the hub is reset. Consequently, this signaling is guaranteed to wakeup and reset this hub only.

11.5.3 Hub behavior during suspend and resume

Hubs must be capable of handling single or multiple resume events that may occur either when the hub is going into suspend or coming out of suspend. This can be best understood with the help of a state machine (Hub Functional State Machine) shown in Figure 11-17. This state machine coordinates handling of suspend and resume operations among a hub and its ports. This is necessary to guarantee reliable and predictable operation of USB suspend and resume operations in all permitted USB topologies in the presence of asynchronous connect, disconnect and remote wakeup events. The device initiating a remote wakeup should see a consistent resume signaling paradigm independent of the state of the hub. The approach outlined below ensures that a resume event will wake up the entire segment of the tree that was previously enabled, up to the first Awake hub with a suspended port. Status at the port indicates that at least one wakeup event occurred to wake up the segment of the tree.

The Hub Functional State Machine contains 6 states: The Awake, Send_Resume, Ds_conn and three Suspend states- Transient, Timed and Low Power Suspend states.

In the USB Host, the Host Port behaves as a downstream port in a hub that is always in the Awake state.

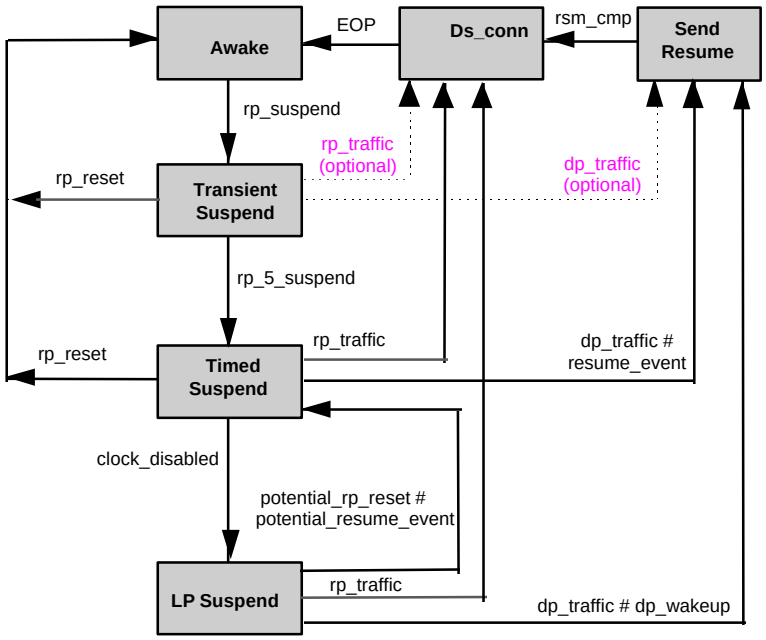

Figure 11-17 Hub Functional State Machine

Awake State

Upon detection of reset on the root port (> 2.5 usec of SE0 detected), a hub reset, or completion of sending resume signaling, the hub enters the Awake state. While in the Awake state, normal upstream and downstream connectivity may be established. The hub remains in the Awake state, until at least 3 ms of IDLE signaling is detected on the root port (rp_suspend). Within 4 ms of IDLE being present on the hub's root port, the Transient Suspend state must be entered. While in the Awake state, resume "K" signaling detected on suspended ports is treated as an indication to enter the selective resume sequence.

Suspend States

While in any of the three Suspend states, (Transient Suspend, Timed Suspend and Low Power Suspend), no connectivity is established, but detection of traffic causes entry to the Send_Resume or the DS_Conn state. On the root port, a "K" detected is considered as traffic, (rp_traffic). On downstream ports, any "K" detected on an enabled port is considered as traffic (dp_traffic) and any "K" detected on a Suspended port is considered a wakeup event (dp_wakeup).

The Transient Suspend state is present to allow all devices and hubs on USB to enter their Suspend states. In this state, no upstream or downstream connectivity is permitted. After an additional 2 ms in this state, the "Timed Suspend" state is entered. The hub **is permitted but not required to** transition to the "Send Resume" state if traffic is detected on a downstream port, and to transition to the "Ds_conn" state if traffic is detected on the root port. Because the host or device sending the traffic is expected to wait 5 ms before sending resume signaling, additional checking by the hub is not considered necessary. A "K" detected on a suspended port causes the selective resume sequence to be initiated.

In the "Timed Suspend" state the hub has access to a clock to time events such as connect, disconnect and reset events. The accuracy of this clock may be different from the clock in the Awake, Resume and Transient Suspend states. From this state, traffic detected on a downstream port (dp_traffic) or the root port (rp_traffic) causes entry to the "Send_Resume" state or the DS_conn state, respectively, within 100 µsec (t_1). Detection of a resume event, (see below) also causes entry to the "Send Resume" state. If clocks are disabled (clock_disabled) to achieve lower power, the "LP Suspend" state is entered.

The following are considered a resume_event:

- Detection of a “K” on a Suspended Port - (dp_wakeup)
- The C_PORT_SUSPEND field of any downstream port is currently set as a result of the completion of the resume signaling sequence that has caused a change from the port suspend to port enabled state. This field is set 3 ms after resume signaling on the port is finished.
- *(Permitted, not required)* Port in process of the selective resume signaling sequence where the C_PORT_SUSPEND field has not been set yet.
- When the Remote Wakeup feature is enabled the following also are considered resume events:
    - the C_PORT_CONNECTION field is currently set as a result of an unacknowledged Connect or Disconnect event.
    - *(Permitted, not required)* Any other bit set in a Port Change field (C_PORT_OVER_CURRENT, C_PORT_RESET, C_PORT_ENABLE)
    - *(Permitted, not required)* A Port is in the process of performing a Port Reset
    - *(Permitted, not required)* Any bit set in the Hub Change field (C_HUB_OVER_CURRENT, C_HUB_LOCAL_POWER)
    - Potential resume events which would result in the eventual setting of these bits. (The corresponding *permitted, not required* options would apply to these events)

In the “LP Suspend” state asynchronous detection of line states on ports is necessary since no clocks are available. Traffic detected on the root port (rp_traffic) causes entry to the “Ds_conn” state. A “K” detected on any enabled or suspended downstream port (dp_traffic or dp_wakeup) causes entry to the Send Resume state. These transitions must occur within 100 usec (t1- see section 11.5.1). Should a potential reset be detected on the root port (rp_reset) or a potential resume event occur the Timed Suspend State is entered. A potential root port reset occurs on detection of a transition to SE0, since when this is timed a reset event may be detected. Similarly a potential resume event is one which would cause changes in the C_PORT fields gated by the Remote Wakeup feature as listed above. For example, this could be a transition to SE0 on a downstream port, since when this is timed a disconnect event may be detected. Detection of a “K” on a suspended port (dp_wakeup) is not considered a potential resume event.

Send_Resume State

While in the “Send Resume” state, resume “K” signaling is sent upstream, on all enabled (Full Speed and Low Speed) downstream ports and on the port initiating the resume signaling. When propagating resume signaling detected on a downstream device (entrance condition was dp_traffic or dp_wakeup), exit from this state must not occur before 1 ms of resume signaling. Before leaving the Send_Resume state, the Hub must have stable Power and clocks.. The “Send Resume” State must persist for no longer than 15 ms. Upon exit from the “Send Resume” state, the bus is floated (No LS EOP is sent) and the “Ds_conn” state is entered.

While in the Send_Resume state, resume “K” signaling detected on suspended ports is treated as an indication to reflect resume “K” signaling within 100 μsec and to enter the selective resume sequence. Exit from the Send_Resume state (the resume_cmp condition) must occur no earlier than 1 ms and no later than 15 ms from when it was entered. The 15 ms maximum includes the time to enable clocks if clocks were disabled.

Ds_conn State

While in the “Ds_conn” state, signaling received on the root port is passed downstream to all enabled downstream ports, including Low Speed Ports. Exit from this state occurs upon detection of an EOP on the root port.

A few illustrative cases are reviewed here:

Case 1:

The hub functional state is Awake and the port state is Suspend. The port receives resume "K" signaling and starts the selective resume sequence. While that is going on, 3 ms of Idle is detected on the hub's root port and the hub enters the "Transient Suspend" state. Within the next 2 ms thereafter, the selective resume sequence on the port completes. At the end of the 2 ms, the Timed Suspend state is entered and then immediately the "Send Resume" state is entered due to the pending resume_event.

Note that in this case, if one or more ports has an event pending the results are the same.

Case 2:

The hub functional state is "LP Suspend" and the hub has an enabled and a suspended port. Resume "K" signaling is detected on the enabled port, the hub wakes up and enters the "Send Resume" state within 100 usec (t1). After at least 1 ms, and a maximum of 15 ms, the hub is powered on and clocks are accurate. The Hub enters the Ds_conn state and propagates the signaling at the root port to the downstream enabled ports. At the EOP which marks the end of the resume signaling at the root port, the hub enters the Awake state..

While in the "Awake" state, resume "K" signaling is detected on the "Suspended" port. This causes the port to be selectively resumed by the hub driving resume signaling for at least 20 ms followed by an EOP. So at the end of at least 23 ms, the Suspend Status change bit is set.

Case 3:

The hub functional state is "LP Suspend" and there is an enabled and suspended port. Resume "K" signaling is detected on the enabled port, the hub initiates its wake up sequence and enters the "Send Resume" state within 100 usec. While in the "Send Resume" state, resume "K" signaling is detected on the "Suspended" port. This causes the selective resume sequence to be initiated. The hub drives resume signaling down the port till it is ready to time events. After this point the hub continues to drive resume signaling for at least 20 ms and the hub is in the Awake state. This guarantees that a device will see at least 20 ms of continuous resume signaling. This provides sufficient time for the device to complete its wakeup sequence.

11.6 USB Hub Reset Behavior

A USB hub must be able to generate a per port reset via a host request and accept reset signaling on its root port. The following sections describe hub reset behavior and its interactions with resume, attach detect, and power-on.

11.6.1 Hub Receiving Reset on Root Port

Reset signaling to a hub is defined only in the downstream direction which is at the hubs' root port. An awake hub may start its reset sequence if it detects 2.5 μs or more of continuous SE0 signaling and must complete its reset sequence no later than 5.5 μs of continuous SE0. The 2.5 μs lower limit is set by a need to prevent low speed EOP strobes (which are up to 1.3 μs in length) from being interpreted as reset. A suspended hub must interpret the start of reset edge as a resume signaling event and begin its wake-up sequence. The hub must be awake and have completed reset no later than 10 ms after the completion of reset signaling

After completion of reset a hub controller is in the following state:

- Hub controller default address is 0
- Hub control bits set to default values
- Hub repeater is in the WFDSOP state

- All downstream ports in Powered Off state (power-switched hubs)
- All downstream ports in Disconnected state (non power-switched hubs)

If a bus contains hubs with power-switched ports, the host reset is not guaranteed to propagate all the way downstream. The host has to guarantee that each tier is reset when it goes through the enumeration process, and the enumeration reset is done on a tier by tier basis. (However, the powered off devices are effectively reset, if they are off long enough, and self-powered devices/hubs below them reset themselves and their downstream ports.)

11.6.2 Per Port Reset

A hub can exercise per-port resets via the SetPortFeature(PORT_RESET) request. This request specifies a downstream port number. In response to a SetPortFeature(PORT_RESET) request, the hub drives an SE0 onto its downstream port for at least 10 ms, returns the bus to the idle (J) state, and then places the port into the enabled state. SetPortFeature(RESET) is an atomic operation; the 10 ms delay between start and end of reset is controlled by the hub. The hub must be able to return to the host the status of the reset request; i.e., whether the reset has completed, so that the host does not have to keep track of elapsed time during the reset operation. The port reset request can be issued to a port in any state; however, no downstream signaling is generated if the reset is issued to a port in the powered off or disconnected states.

Bus state evaluation after port reset is described in sec. 11.2.4. Device attach detection requires that the port in question be power-switched on (if power switching is an option). When a device is attached, a hub can detect an attach via an SE0 to DIFF1 or DIFF0 bus transition. This requires that disabled ports not be driven by the hub while attach detection is being performed. This should not be a problem, as the port will have been disabled and its output drivers floated by detection of the previous detach event. The host can determine the device's speed by examining whether D+ or D- is pulled high.

Before a port to which a device has been connected can be enabled it must be assured that the device has been reset. Since it is not possible to rely on loss of Vbus, caused by a disconnect event, to reset the device, a port must be reset before being enabled. This is performed via an atomic SetPortFeature(PORT_RESET) request which issues SE0 reset signaling and then enables the port. USB devices, including hubs, must be able to respond to a host access no later than 10 ms after reset is de-asserted.

11.6.3 Power Bringup and Reset Delays

Since USB components (including hubs) 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 is capable of accepting downstream bus traffic, especially bus reset. When a bus powered device is hot plugged or when a self powered device is powered on, sufficient time must be allotted to guarantee that the host does not attempt to reset the device before it is capable of accepting the reset. A similar restriction applies for resetting a device which is connected to a hub port that is power switched.

Figure 11-18 shows the case where a device is connected to a hub whose port is power switched. For this case two delays need to be taken into consideration. $\Delta t1$ is the amount of time required for the hub's power switch to operate and for V_{BUS} at the downstream port to stabilize. $\Delta t2$ is the amount of time for the device's internal power supply to stabilize and any power-on reset to complete. $\Delta t1$ is a function of the hub's implementation of power switching and may be read by the host via a hub controller command. If a hub implements power switching then $\Delta t1 + \Delta t2$ must elapse before bus reset is applied. If a device is hot plugged to an already powered port then only the $\Delta t2$ parameter is relevant.

$\Delta t2$ is assigned a maximum value of 100 ms; if a device cannot meet this limit then it must 'appear' unconnected to the hub until the device is capable of accepting an SE0 bus reset.

USB devices, including hubs, must power up in such a manner that they do not drive either D+ or D- (except with the pullup resistor) before or during the reset process. This is required so the upstream hub can drive reset downstream and be assured that the downstream device will see the reset signaling

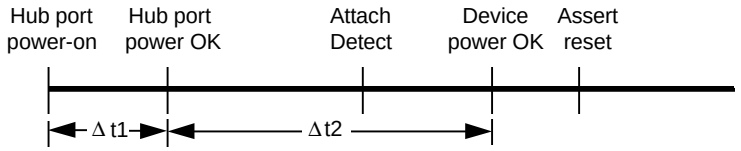

Figure 11-18. Power-on timing

11.7 Hub Power Distribution Requirements

Hubs can supply a specified amount of power to downstream components and are responsible for reporting their power distribution capabilities to the host during enumeration. USB requirements stipulate that generalized legal bus topologies be supported while at the same time preventing power-up of illegal topologies. An illegal power topology is one that violates the power contract established during enumeration.

Hubs may be either locally powered or bus powered, or a combination of the two. For example, a hub may derive power for its SIE and root port pull-up resistors from the bus while obtaining power for its downstream ports from a local power supply. A hub can only supply power in a downstream direction, and must never drive power upstream. A complete discussion of hub power distribution appears in Section 7.2.

Self powered hubs must provide overcurrent protection at the ports, bus powered hubs must have port power switching for their downstream ports. Hubs with power-switched ports are required to power off all downstream ports until the hub is configured. The ports should also be powered off when the hub receives a reset on its root port. Ports may also be switched on and off under host software control. An implementation may provide power switching on a per port basis or have a single switch for all the ports (gang mode power control) or a combination of these two as specified in a power control mask (see Table 11-8). The gang-mode is a logic-OR mode; i.e., all ports in a gang will be

switched on if any of the ports is switched on and all will need to be switched off for any to be switched off. Port reset requests do not affect the status of the power switching for a port. A hub port must be powered on in order to perform connect detection from the downstream direction.

11.7.1 Overcurrent Indication

For reasons of safety, all locally powered hubs must implement current limiting on their downstream ports. Under no conditions may more than 25 VA be drawn from any USB hub port. (The actual overcurrent trip point may be lower than this figure). If an overcurrent condition occurs, even if it is only momentary, it must be reported to the hub controller. This is done via an overcurrent state that is reflected in the hub or port status. The overcurrent detect state is entered on overcurrent detect and cleared by a host request or upon reset. Detection of overcurrent must place all affected ports in the powered off state (if the hub supports power switching) or in the disconnected state (if the hub does not support power switching).. If the overcurrent condition has caused a permanent disconnect of power (such as a blown fuse), the hub must report it upon coming out of reset or power-up.

Overcurrent protection may be implemented over all downstream ports in aggregate, or on a per port basis. The ports may optionally be split into two or more subgroups, each with its own overcurrent protection circuit.

11.8 Hub Endpoint Organization

The Hub Class defines one additional endpoint beyond Endpoint 0, which is required for all devices: the Status Change endpoint. The host system receives port and hub status change notifications through the Status Change endpoint. The Status Change endpoint supports interrupt transfers. If the hub has not detected changes on any of its ports, nor any hub status changes, the hub returns a NAK to requests on the Status Change endpoint. When the hub detects any status change, the hub responds with data describing the entity that changed. Host software driving the hub is responsible for examining the data transferred to determine which entity changed. Hubs are logically organized as shown in Figure 11-19.

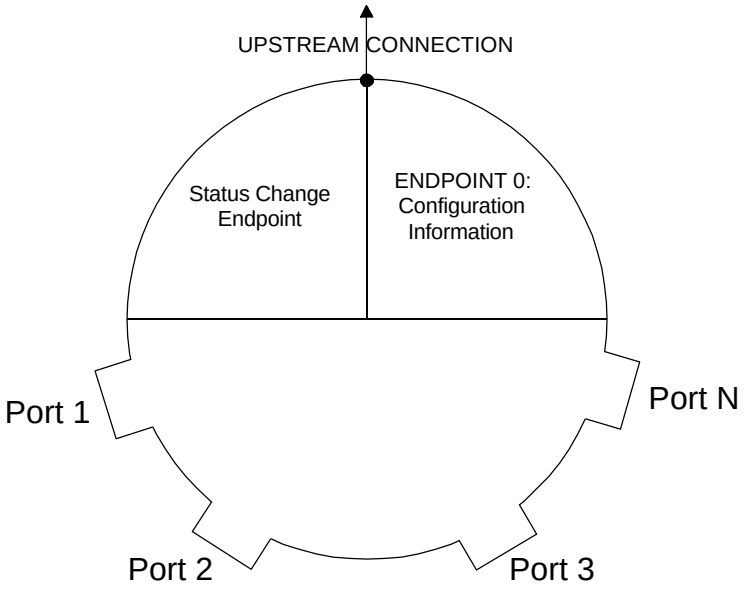

Figure 11-19. Example Hub Organization

11.8.1 Hub Information Architecture and Operation

Hub Descriptors and Hub/Port Status and Control are accessible through the default pipe. When a hub detects a change on a port or when the hub changes its own state, the Status Change endpoint transfers data to the host in the form specified in Section 11.8.3.

USB hubs detect changes in port status. Devices attached to the ports on a hub can cause various hardware events. In addition, host system software can cause changes to a port's state by sending commands to the hub. In all cases, USB hubs report change information for all of the recognized events which currently apply. Recognized events are defined in Table 11-16. The hub continues to report a status change when polled until that particular event has been successfully acknowledged by the host. Using this reporting mechanism, system software determines what changes occurred since the last event reported by the hub. This approach makes it possible to minimize the device state information that system software must carry.

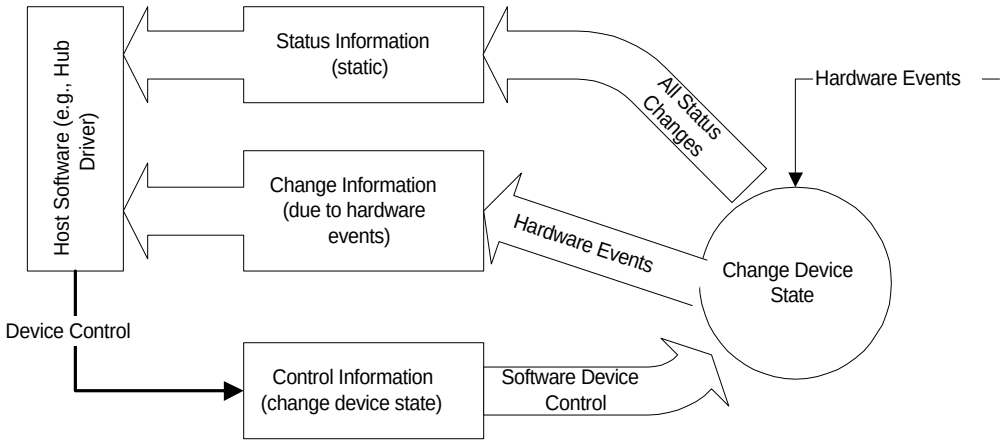

Figure 11-20. Relationship of Status, Status Change, and Control Information to Device States

Host software uses the interrupt pipe associated with the Status Change endpoint to detect changes in hub and port status.

11.8.2 Port Change Information Processing

Hubs report a port's status through port commands on a per-port basis. Host software acknowledges a port change by clearing the change state corresponding to the status change reported by the hub. The acknowledgment clears the change state for that port so future data transfers to the Status Change endpoint do not report the previous event. This allows the process to repeat for further changes (see Figure 11-21.)

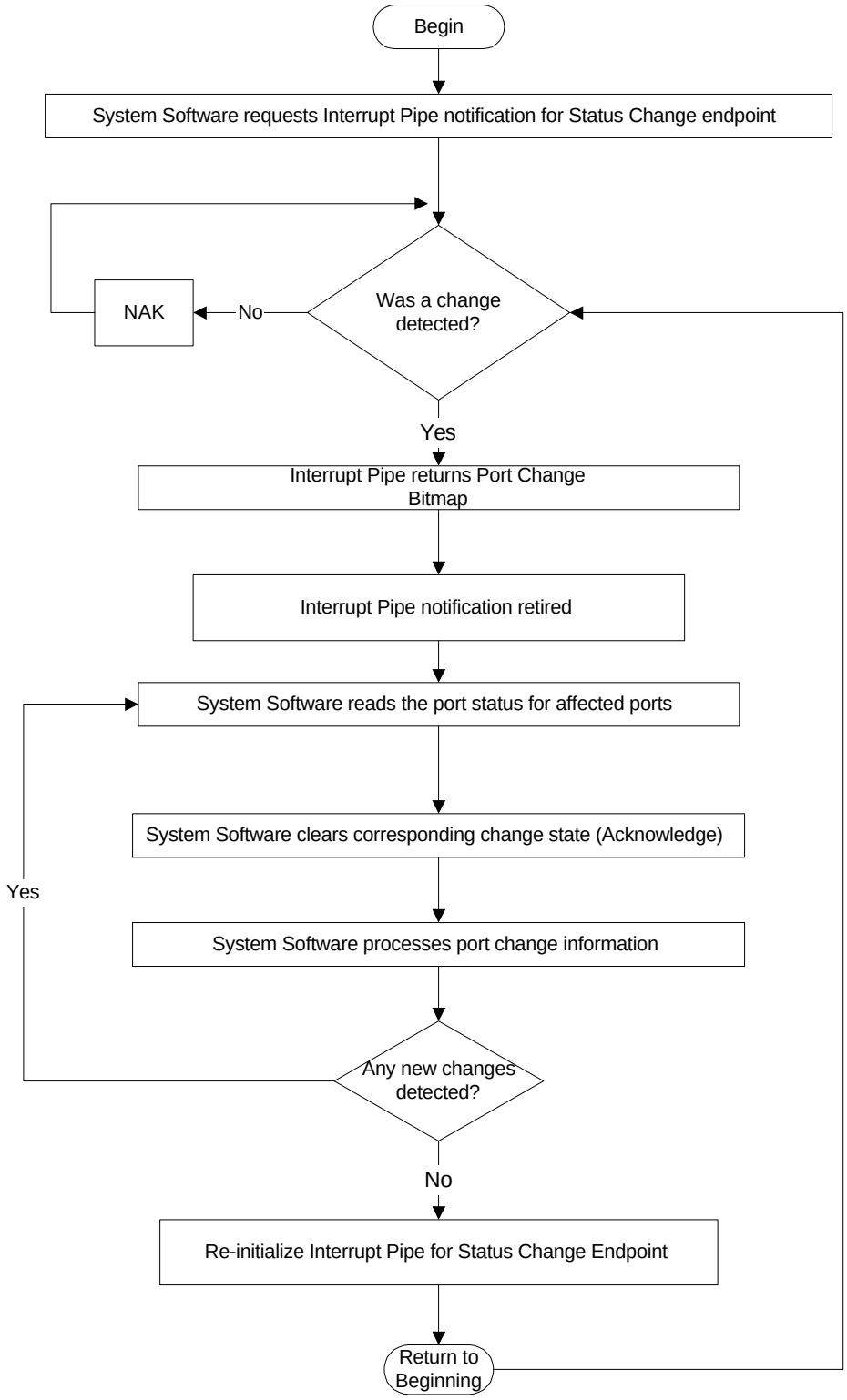

Figure 11-21. Port Status Handling Method

11.8.3 Hub and Port Status Change Bitmap

The Hub and Port Status Change Bitmap, shown in Figure 11-22, indicates whether the hub or a port has experienced a status change. This bitmap also indicates which port(s) have had a change in status. The hub returns this value on the Status Change endpoint. Hubs report this value in byte-increments. That is, if a hub has six ports, it returns a byte quantity and reports a zero in the invalid port number field locations. System software is aware of the number of ports on a hub (this is reported in the hub descriptor) and decodes the Hub and Port Status Change Bitmap accordingly. The hub reports any changes in hub status on bit 0 of the Hub and Port Status Change Bitmap.

The Hub and Port Status Change Bitmap size varies from a minimum size of one byte. Hubs only report as many bits as there are ports on the hub, subject to the byte-granularity requirement (i.e., round up to the nearest byte).

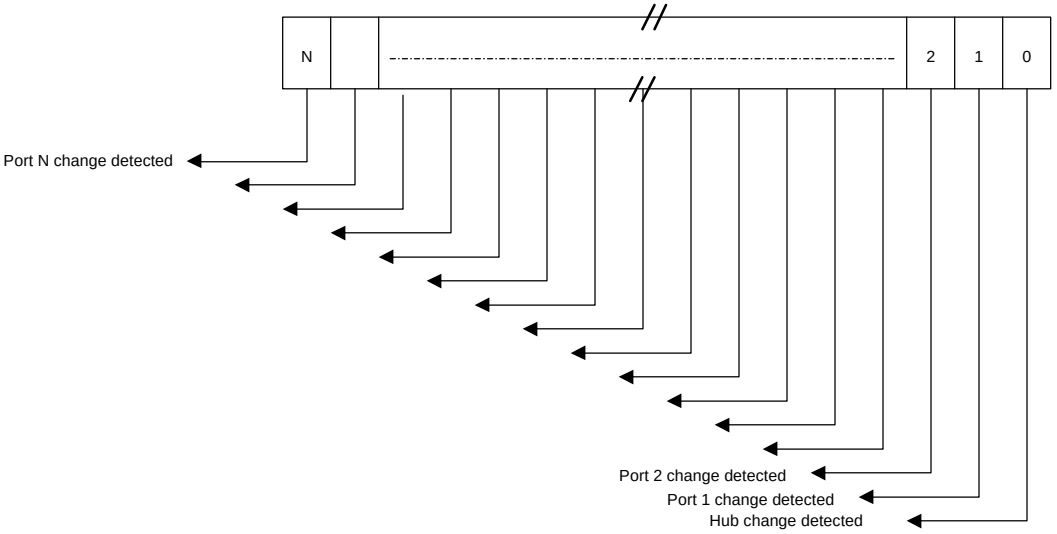

Figure 11-22. Hub and Port Status Change Bitmap

Any time the Status Change endpoint is polled by the host controller and any of the Status Changed bits are non-zero, the Hub and Port Status Change Bitmap is returned. Hubs sample the change at the End of Frame (EOF2) in preparation for a potential data transfer in the subsequent USB frame. If a change was detected, then data will be transferred through the Status Change endpoint in the subsequent USB frame. Figure 11-23 shows the sampling mechanism for hub and port change bits.

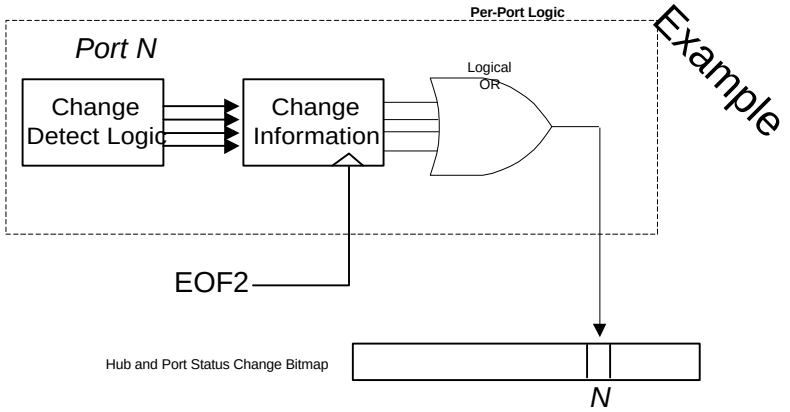

Figure 11-23. Example Hub and Port Change Bit Sampling

11.9 Hub Configuration

Hubs are configured through the standard USB device configuration commands. An unconfigured hub behaves like all other unconfigured devices with respect to power requirements and addressability. Unconfigured hubs do not turn power onto the downstream ports. Configuring a hub enables the Status Change endpoint. System software issues commands to the hub to switch port power on and off at appropriate times.

System software examines hub descriptor information before configuration to determine the hub's characteristics. By examining the hub's characteristics, system software ensures that illegal power topologies are not allowed by not powering on the hub's ports if doing so would violate the USB power topology.

11.10 Hub Port Power Control

Hubs allow port power to be controlled by the host system. As described previously, hubs support per-port or gang-mode power switching on the downstream ports. Switching port power is done via hub commands, defined below. Hubs with per-port power switching may also allow gang-mode power switching by specifying a certain value in a request field in the port power control request (refer to the SetPortFeature(PORT_POWER) request definition in Section 11.12.2.9).

Hubs may wish to mask the gang-mode power control for certain ports. This allows a hub to independently control the power switching for certain ports, regardless of the general port switching characteristics. For example, consider a hub with gang-mode port power switching that has a permanently attached device on a port (an "embedded port"). If the Port Power Control Mask field for the embedded port indicates that gang-mode power switching is masked, any hub commands that control the ports in gang-mode will not affect the embedded port.

11.11 Descriptors

Hub descriptors are derived from the general USB device framework. Hub descriptors define a hub device and the ports on that hub. The host accesses hub descriptors through the hub's default pipe.

The USB Specification (refer to Chapter 9) defines the following descriptors:

- Device
- Configuration
- Interface
- Endpoint
- String (optional)

The hub class defines additional descriptors (refer to Section 0). In addition, vendor-specific descriptors are allowed in the USB device framework. Hubs support standard USB device commands as defined in Chapter 9.

11.11.1 Standard Descriptors

The hub class pre-defines certain fields in standard USB descriptors. Other fields are either implementation-dependent or not applicable to this class.

Note: For the descriptors and fields shown below, the bits in a field are organized in a Little-Endian fashion. That is, bit location 0 is the least significant bit, and bit location 7 is the most significant bit of a byte value.

Device Descriptor

bDeviceClass	=		Implementation Dependent (0x00 or HUB_CLASSCODE) This field can contain either a zero or HUB_CLASSCODE. A zero in this field indicates that the device's interface(s) operate independently and each interface descriptor's bInterfaceClass must be examined for the class code. If this field contains HUB_CLASSCODE, then this device has ONLY one interface and that interface conforms exactly to the USB hub class definition. However, in this case, the interface descriptor's bInterfaceClass field must also contain HUB_CLASSCODE.
bDeviceSubClass	=	0	
wMaxPacketSize0	=	8 bytes	

Interface Descriptor

bNumEndpoints	=	1
bInterfaceClass	=	HUB_CLASSCODE
bInterfaceSubClass	=	0
bInterfaceProtocol	=	0

Configuration Descriptor

MaxPower	=	The maximum amount of bus power this hub will consume in this configuration.

Endpoint Descriptor (for Status Change Endpoint)

bEndpointAddress	=	Implementation dependent; Bit7: Direction = In (1)
wMaxPacketSize	=	Implementation dependent
bmAttributes	=	Transfer Type = Interrupt (0b00000111)
bInterval	=	0xFF (Maximum allowable interval)

The hub class driver retrieves a device configuration from host system software using the GetDescriptor device request. The first endpoint descriptor returned by GetDescriptor request is, by specification, the Status Change endpoint descriptor. Hubs may define additional endpoints beyond the minimum required by this class definition. However, hubs conforming to this class standard always return the Status Change endpoint as the first endpoint descriptor in the standard interface.

11.11.2 Class-specific Descriptors

11.11.2.1 Hub Descriptor

Table 11-8. Hub Descriptor

Offset	Field	Size	Description
0	*bDescLength*	1	Number of bytes in this descriptor, including this byte.
1	*bDescriptorType*	1	Descriptor Type, value: 0x29 for Hub Descriptor.
2	*bNbrPorts*	1	Number of downstream ports that this hub supports.

Offset	Field	Size	Description
3	*wHubCharacteristics*	2	D1..D0: Power Switching Mode 00 - Ganged power switching (all ports' power at once) 01 - Individual port power switching 1X - No power switching (ports always powered on when hub is on and off when hub is off). D2: Identifies a Compound Device 0 - Hub is not part of a compound device 1 - Hub is part of a compound device D4..D3: Over-current Protection Mode 00 - Global Over-current Protection. The hub reports over-current as a summation of all ports' current draw, without a breakdown of individual port over-current status. 01 - Individual Port Over-current Protection. The hub reports over-current on a per-port basis. Each port has an over-current indicator. 1X -No Over-Current Protection. This option is only allowed for bus-powered hubs that do not implement over-current protection. D15..D5: Reserved
5	*bPwrOn2PwrGood*	1	Time (in 2 ms intervals) from the time power on sequence begins on a port until power is good on that port. System software uses this value to determine how long to wait before accessing a powered-on port.
6	*bHubContrCurrent*	1	Maximum current requirements of the hub controller electronics in mA.

Offset	Field	Size	Description
7	*DeviceRemovable*	Variable depend-ing on number of ports on hub	Indicates if a port has a removable device attached. If a non-removable device is attached to a port, that port will never receive an insertion change notification. This field is reported on byte-granularity. Within a byte, if no port exists for a given location, the field representing the port characteristics returns “0”. Bit definition: 0 - Device is removable 1 - Device is not removable (permanently attached) This is a bitmap corresponding to the individual ports on the hub: Bit 0: Reserved for future use Bit 1: Port 1 Bit 2: Port 2 Etc. Bit *n*: Port *n* (implementation dependent, up to a maximum of 255 ports).

Offset	Field	Size	Description
Variable	*PortPwrCtrlMask*	Variable depending on number of ports on hub	Indicates if a port is not affected by a gang-mode power control request. Ports that have this field set always require a manual SetPortFeature(PORT_POWER) request to control the port's power state. Bit definition: 0 - Port does not mask the gang-mode power control capability. 1 - Port is not affected by gang-mode power commands. Manual commands must be sent to this port to turn power on and off. This is a bitmap corresponding to the individual ports on the hub: Bit 0: Reserved for future use. Bit 1: Port 1 Bit 2: Port 2 Etc. Bit *n*: Port *n* (implementation dependent, up to a maximum of 255 ports).

11.12 Requests

11.12.1 Standard Requests

Table 11-9. Hub Responses to Standard Device Requests

bRequest	Hub Response
CLEAR_FEATURE	Standard
GET_CONFIGURATION	Standard
GET_DESCRIPTOR	Standard
GET_INTERFACE	Optional. Hubs only required to support one interface
GET_STATUS	Standard
SET_ADDRESS	Standard
SET_CONFIGURATION	Standard
SET_DESCRIPTOR	Optional
SET_FEATURE	Standard
SET_INTERFACE	Optional. Hubs only required to support one interface
SYNCH_FRAME	Optional. Hubs are not required to have isochronous endpoints.

11.12.2 Class-specific Requests

The hub class defines requests to which all hubs must respond.

Table 11-10. Hub Class Requests

Request	bmRequestType	bRequest	wValue	wIndex	wLength	Data
ClearHubFeature	00100000B	CLEAR_ FEATURE	Feature Selector	Zero	Zero	None
ClearPortFeature	00100011B	CLEAR_ FEATURE	Feature Selector	Port	Zero	None
GetBusState	10100011B	GET_ STATE	Zero	Port	One	Per Port Bus State
GetHubDescriptor	10100000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
GetHubStatus	10100000B	GET_ STATUS	Zero	Zero	Four	Hub Status and Change Indicators
GetPortStatus	10100011B	GET_ STATUS	Zero	Port	Four	Port Status and Change Indicators
SetHubDescriptor	00100000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
SetHubFeature	00100000B	SET_ FEATURE	Feature Selector	Zero	Zero	None
SetPortFeature	00100011B	SET_ FEATURE	Feature Selector	Port	Zero	None

Table 11-11. Hub Class Request Codes

bRequest	Value
GET_ STATUS	0
CLEAR_ FEATURE	1
GET_STATE	2
SET_ FEATURE	3
reserved for future use	*4-5*
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7

The following are the valid feature selectors for the hub class. See GetHubStatus and GetPortStatus for a description of the features.

Table 11-12. Hub Class Feature Selectors

	Recipient	Value
C_HUB_LOCAL_POWER	Hub	0
C_HUB_OVER_CURRENT	Hub	1
PORT_CONNECTION	Port	0
PORT_ENABLE	Port	1
PORT_SUSPEND	Port	2
PORT_OVER_CURRENT	Port	3
PORT_RESET	Port	4
PORT_POWER	Port	8
PORT_LOW_SPEED	Port	9
C_PORT_CONNECTION	Port	16
C_PORT_ENABLE	Port	17
C_PORT_SUSPEND	Port	18
C_PORT_OVER_CURRENT	Port	19
C_PORT_RESET	Port	20

11.12.2.1 Clear Hub Feature

This request resets a value reported in the hub status.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	CLEAR_ FEATURE	Feature Selector	Zero	Zero	None

Clearing a feature disables that feature; refer to Table 11-12 for the feature selector definitions. If the feature selector is associated with a change indicator, clearing that indicator acknowledges the change. Both C_HUB_LOCAL_POWER and C_HUB_OVER_CURRENT may be acknowledged using this request.

11.12.2.2 Clear Port Feature

This request resets a value reported in the port status.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100011B	CLEAR_ FEATURE	Feature Selector	Port	Zero	None

The port number must be a valid port number for that hub, greater than zero.

Clearing a feature disables that feature; refer to Table 11-13 for the feature selector definitions. If the feature selector is associated with a change indicator, clearing that indicator acknowledges the change. Changes in connection, enable, suspend, reset, and over-current status are acknowledged using this request.

Clearing the PORT_SUSPEND feature causes a host-initiated resume on the specified port. Clearing the PORT_ENABLE feature causes the port to be disabled. Clearing the PORT_POWER feature causes the port to be powered off, subject to the constraints due to the hub's method of power switching. If a hub uses gang power switching, all ports must be requested to power off before any of the ports actually power off.

11.12.2.3 Get Bus State

This is an optional per-port diagnostic request which reads the bus state value, as sampled at the last EOF2.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	GET_ STATE	Zero	Port	One	Per Port Bus State

The port number must be a valid port number for that hub, greater than zero.

Hubs may implement an optional diagnostic aid to facilitate system debug. Hubs implement this aid through this optional request. This diagnostic feature provides a glimpse of the USB bus state as sampled at the last EOF2 sample point.

Hubs that implement this diagnostic feature should store the bus state at each EOF2 state, in preparation for a potential request in the following USB frame.

The data returned is bit-mapped in the following manner. The value of the D- signal is returned in the field in bit 0. The value of the D+ signal is returned in the field in bit 1. Bits 2-7 are reserved for future use and are reset to zero.

Hubs that do not support this request respond with a stall.

11.12.2.4 Get Hub Descriptor

This request returns the hub descriptor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero	Descriptor Length	Descriptor

The GetDescriptor request for the hub class descriptor follows the same usage model as that of the standard GetDescriptor request (refer to Chapter 9). The standard hub descriptor is denoted by using the value bDescriptorType defined in Section 11.11.2.1. All hubs are required to implement one hub descriptor, with descriptor index zero.

11.12.2.5 Get Hub Status

This request returns the current hub status and the states that have changed since the previous acknowledgment.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100000B	GET_ STATUS	Zero	Zero	Four	Hub Status and Change Indicators

The first word of data contains *wHubStatus* (refer to Table 11-13). The second word of data contains *wHubChange* (refer to Table 11-14).

The fields returned are organized in such a way to allow system software to determine which states have changed. The bit locations in the *wHubStatus* and *wHubChange* fields correspond in a one-to-one fashion where applicable.

Local power and overcurrent changes are acknowledged using the ClearHubFeature request.

Table 11-13. Hub Status Field, *wHubStatus*

BIT	DESCRIPTION
0	**Local Power Status:** This is the state of the local power supply. This field only applies to self-powered hubs whose USB Interface Engine (SIE) is bus-powered or hubs that support either self-powered or bus-powered configurations. This field is returned as a result of a change to the hub's power source. This field reports whether local power is good. This field allows system software to determine the reason for the removal of power to devices attached to this hub or to react to changes to the local power supply state. If the hub does not support this feature, this field is RESERVED and follows the definition of the RESERVED bits below. This field reports the power status for the SIE and the remainder of the hub. 0 = Local power supply good 1 = Local power supply lost (inactive)
1	**Over-Current Indicator:** This field only applies to hubs that report over-current conditions on a global hub basis (as reported in the **Hub Descriptor**). If the hub does not report over-current on a global hub basis, this field is RESERVED and follows the definition of the RESERVED bits below. This field indicates that the sum of all the ports' current has exceeded the specified maximum and power to all the ports has been shut off. For more details on Over-Current protection, see Section 7.2.1.2.1. This field indicates an over-current condition due to the sum of all ports' current consumption. 0 = All power operations normal 1 = An over-current condition caused power to the ports to be shut off. This bit will remain set until power is restored to the ports AND the port power consumption is within acceptable limits.
2-15	**Reserved** These bits return 0 when read.

Table 11-14. Hub Change Field, *wHubChange*

BIT	DESCRIPTION
0	**Local Power Status Change:** (C_HUB_LOCAL_POWER) This corresponds to Local Power Status, Bit 0 above. This field only applies to locally-powered (i.e., self-powered) hubs whose USB Interface Engine (SIE) is bus-powered, or hubs that support either self-powered or bus-powered configurations. This field is returned as a result of a change to the hub's power source. If the hub does not support this feature, then this field is RESERVED and follows the definition of the RESERVED bits below. This field reports whether a change has occurred to the local power status. 0 = No change has occurred on Local Power Status 1 = Local Power Status has changed
1	**Over-Current Indicator Change**: (C_HUB_OVER_CURRENT) This corresponds to Over-Current Indicator, Bit 1 above. This field only applies to hubs that report over-current conditions on a global hub basis (as reported in the Hub Descriptor). If the hub does not report over-current on a global hub basis, this field is RESERVED and follows the definition of the RESERVED bits below. This field reports whether a change has occurred to the Over-Current Indicator. This field is only set if an over-current condition has occurred (i.e., acknowledgment of this change by system software will not cause another change to be reported). 0 = No change has occurred on the Over-Current Indicator 1 = Over-Current Indicator has changed . An over-current condition caused power to the ports to be shut off, or the port power was restored. This bit will remain set until cleared by a CLEAR_ FEATURE (C_HUB_OVER_CURRENT) command.
2-15	**Reserved** These bits return 0 when read.

11.12.2.6 Get Port Status

This request returns the current port status and the states that have changed since the previous acknowledgment.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	GET_STATUS	Zero	Port	Four	Port Status and Change Indicators

The port number must be a valid port number for that hub, greater than zero.

The first word of data contains *wPortStatus* (refer to Table 11-15). The second word of data contains *wPortChange* (refer to Table 11-16).

The fields returned are organized in such a way to allow system software to determine which states have changed. The bit locations in the *wPortStatus* and *wPortChange* fields correspond in a one-to-one fashion where applicable.

Table 11-15. Port Status Field, *wPortStatus*

BIT	DESCRIPTION
0	**Current Connect Status:** (PORT_CONNECTION) This field reflects whether or not a device is currently connected to this port. This value reflects the current state of the port, and may not correspond directly to the event that caused the Insertion Status Change (Bit 0 in below) to be set. 0 = No device is present on this port 1 = A device is present on this port **NOTE: This field is always 1 for ports that have non-removable devices attached.**
1	**Port Enabled/Disabled:** (PORT_ENABLE) Ports can be enabled by host software only. Ports can be disabled by either a fault condition (disconnect event or other fault condition, including an over-current indication) or by host software. This bit is asserted while port state is enabled and may be asserted while port state is suspended (see Section 11.2.3). 0 = Port is disabled 1 = Port is enabled
2	**Suspend:** (PORT_SUSPEND) This field indicates whether or not the device on this port is suspended. Setting this field causes the device to suspend by not propagating bus traffic downstream. Resetting this field causes the device to resume. Bus traffic cannot be resumed in the middle of a bus transaction If the device itself is signaling a resume, this field will be cleared by the hub. 0 = Not suspended 1 = Suspended
3	**Over-Current Indicator:** (PORT_OVER_CURRENT) This field only applies to hubs that report over-current conditions on a per-port basis (as reported in the Hub Descriptor). If the hub does not report over-current on a per-port hub basis, this field is RESERVED and follows the definition of the RESERVED bits below. This field indicates that the device attached to this port has drawn current that exceeds the specified maximum and this port's power has been shut off. Port power shutdown is also reflected in the Port Power field above. For more details, see Section 7.2.1.2.1. This field indicates an over-current condition due to the device attached to this port. 0 = All power operations normal for this port. 1 = An over-current condition caused power to this port to be shut off. This bit will remain set until power is restored to the ports AND the port power consumption is within acceptable limits.
4	**Reset**: (PORT_RESET) This field is set when the host wishes to reset the attached device. It remains set until the reset signaling is turned off by the hub and the reset status change field is set. 0 = Reset signaling not asserted 1 = Reset signaling asserted
5-7	**Reserved** These bits return a "0" when read.
8	**Port Power:** (PORT_POWER) This field reflects a port's power state. Since hubs can implement different methods of port power switching, the meaning of this field varies depending on the type of power switching used. The device descriptor reports the type of power switching implemented by the hub. Hubs do not provide any power to their ports until they are in the configured state. 0 = This port is powered OFF 1 = This port is powered ON **NOTE: Hubs that do not support power switching always return a 1 in this field.**
9	**Low Speed Device Attached:** (PORT_LOW_SPEED) This is only relevant if a device is attached. 0 = Full Speed device attached to this port 1 = Low speed device attached to this port
10-15	**Reserved** These bits return 0 when read.

Table 11-16. Port Change Field, *wPortChange*

BIT	DESCRIPTION
0	**Connect Status Change:** (C_PORT_CONNECTION) Indicates a change has occurred in the port's Current Connect Status. The hub device sets this field for any changes to the port device connect status, even if system software has not cleared a connect status change.[1] This bit may optionally be set when the over current status change bit is set. 0 = No change has occurred on Current Connect Status 1 = Current Connect Status has changed **NOTE: For ports that have non-removable devices attached, this field is set only after a RESET condition to indicate to system software that a device is present on this port.**
1	**Port Enable/Disable Change:** (C_PORT_ENABLE) This field is only activated when a change in the port's enable/disable status was detected due to hardware changes. This bit may optionally be set when other change status bits are set.. This field is not set if system software caused a port enable/disable change. 0 = No change has occurred on Port Enabled/Disabled status 1 = Port Enabled/Disabled status has changed
2	**Suspend Change:** (C_PORT_SUSPEND) This field indicates a change in the host-visible power state of the attached device. It indicates the device has transitioned out of the suspend state. Going into the suspend state will not set this field. The Suspend Change field is only set when the entire resume process has completed. That is, the hub has ceased signaling resume on this port and 3 ms have passed to allow the device to resynch to SOF. In a suspended hub this bit may be set as soon as resume signaling is reflected downstream. 0 = No change 1 = Resume complete
3	**Over-Current Indicator Change:** (C_PORT_OVER_CURRENT) This field only applies to hubs that report over-current conditions on a per-port basis (as reported in the Hub Descriptor). If the hub does not report over-current on a per-port hub basis, then this field is RESERVED and follows the definition of the RESERVED bits below. This field reports whether a change has occurred to the port Over-Current Indicator. 0 = No change has occurred on Over-Current Indicator 1 = Over-Current Indicator has changed . An over-current condition caused power to this port to be shut off, or the port power was restored. This bit will remain set until cleared by a CLEAR_ FEATURE (C_PORT_OVER_CURRENT) command.
4	**Reset Change:** (C_PORT_RESET) This field is set when reset processing on this port is complete. As a result of completing reset processing, the enabled status of the port is also set and the suspend change field reset. 0 = No change 1 = Reset Complete
5-15	**Reserved** These bits return 0 when read.

[1] If, for example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be "setting" an already-set bit (i.e., the bit will remain set). However, the hub will transfer the change bit only once when the host controller requests a data transfer to the Status Change endpoint. System software is responsible for determining state change history in such a case.

11.12.2.7 Set Descriptor

This request overwrites the hub descriptor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero	Descriptor Length	Descriptor

The SetDescriptor request for the hub class descriptor follows the same usage model as that of the standard SetDescriptor request (refer to Chapter 9). The standard hub descriptor is denoted by descriptor type zero. All hubs are required to implement one hub descriptor, with descriptor index zero.

This request is optional. This request writes data to a class-specific descriptor. The host provides the data that is to be transferred to the hub during the data transfer phase of the control transaction. This request writes the entire hub descriptor at once.

Hubs must buffer all the bytes received from this request to ensure that the entire descriptor has been successfully transmitted from the host. Upon successful completion of the bus transfer, the hub updates the contents of the specified descriptor.

Hubs that do not support this request respond with a stall.

11.12.2.8 Set Hub Feature

This request sets a value reported in the hub status.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_ FEATURE	Feature Selector	Zero	Zero	None

Setting a feature enables that feature; refer to Table 11-12 for the feature selector definitions. Change indicators may not be acknowledged using this request.

11.12.2.9 Set Port Feature

This request sets a value reported in the port status.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100011B	SET_ FEATURE	Feature Selector	Port	Zero	None

The port number must be a valid port number for that hub, greater than zero.

Setting a feature enables that feature; see for the feature selector definitions. Change indicators may not be acknowledged using this request.

Setting the PORT_SUSPEND feature causes bus traffic to cease on that port and, consequently, the device to suspend. Setting the reset feature (PORT_RESET) causes the hub to signal reset on that port. When the reset signaling is complete, the hub sets the C_PORT_RESET change indicator and immediately enables the port. Refer to Section 11.6.2 for a complete discussion of host initiated reset behavior.