DEVELOPMENT OF A UNIVERSAL SERIAL BUS HUB WITH PROTECTION CIRCUITS

MOHD RAZUAN BIN MOHAMED SAID

A thesis submitted in fulfillment of the requirements for the award of the degree of Bachelor of Electrical Engineering (Electronics)

Faculty of Electrical and Electronics Engineering
Universiti Malaysia Pahang

MAY 2008
Objective of this project is to design, construct and develop a Universal Serial Bus (USB) hub with protection circuits. There are two types of USB hub according to their supplied power that is self powered USB hub and bus powered USB hub. Self powered USB hub means that USB hub get their power from power supply, and bus powered USB hub means that USB hub get their power from personal computer. This project only develops one type of USB hub which is self powered USB hub. In order to wider the scopes of this project, this project is added with powered circuits to make it more firm and hard to be blow out. The most important thing in designing this project is to manage input power and supply power for each port. Four ports are chose to be implementing in the circuits to be connecting into one personal computer. This project also will discuss the result in connecting four computer devices onto one personal computer using final and complete developed gadget
CHAPTER 1

INTRODUCTION

This chapter will discuss on the term Universal Serial Bus (USB) and USB hub. The problem statement which contributes to the creation and development of this project, overview, objective and scope of this project is also presented in this chapter.

1.1 Overview

It seems nearly every electronic gadget manufactured today makes use of the ever-handly USB port. Early computers featuring USB normally had only one or two ports located inconveniently at the back of the case. Newer computers feature several built-in USB ports, and many cases now place two or more of these ports in front. Even so, additional ports are often needed, and having them conveniently accessible makes all the difference. The USB hub does just that.[1]

A USB hub is a small, light unit with multiple ports for plugging in USB devices. It is commonly connected to a USB port located on the back of a desktop computer by using an extension cable. Once the hub is plugged in, you can set it wherever is convenient, avoiding the hassle of accessing the rear of the system. A USB hub is also great for laptops with only one or two ports. Most hubs can support up to 127 devices.[1]
A self-powered USB hub can be used to connect digital cameras, card readers, keyboards, mice, MP3 players, memory sticks and many other handheld USB devices. For more robust components such as external drives, printers, scanners or fax machines, an AC-powered USB hub is a better choice. Some of the AC-powered hubs come with an AC-adapter, while others have the capability, but require the separate purchase of an adapter. [1]

Another feature to look for is 1.1 or 2.0 compliancy. This refers to the two versions of USB technology. USB 1.1 is capable of data transfer speeds up to 12 megabits per second (mbps), while USB 2.0 can transfer data at 480 mbps, 40x faster. Initial USB devices were engineered to use USB 1.1, while later devices took advantage of the newer 2.0 compliancy. [1]

A USB hub that supports 2.0 is often backwards compatible, supporting 1.1 devices as well. The hub automatically detects and runs at the fastest rate the device will support. A USB hub that supports 2.0 cannot "push" a 1.1 device to run faster than its design. Conversely, a hub that only supports 1.1 may or may not support a 2.0 device, but if it does, it will slow it down to 12 mbps — the fastest speed the hub supports. [1]

Virtually all USB devices are plug 'n' play, or hot-swappable, but it's wise to be conservative with this feature when using an external hard drive with a USB port. Data could be lost due to software bugs or if the drive is unplugged while busy. [1]

A USB hub is an inexpensive, handy addition to any system, especially useful with laptops that normally have too few native USB ports and older systems that have rear ports. A USB hub can be purchased in a four-port model, a seven-port model, or greater. Multiple hubs can be used for scalable growth. [1]
1.2 Objective

The objectives of this project are stated below:

1.2.1 To transmit and receive data using USB controller.

This objective is desire a fully functional USB hub as the final product. This is the main objectives of the entire project. The USB hub also must be able to transmit and receive data between at least two devices and one computer.

1.2.2 To develop a protection circuits for USB hub.

The objective is to provide a protection circuits to the USB hub. Their first priority is to protect the USB controller and components that being use in transmitting and receiving data. The circuits should not fragile and must not require any circuit to protect it.

1.2.3 To make this project as reference to others in the future.

After a series of research, I have discovered that there is only limited information about Universal Serial Bus out there. The only mass information is about how to make and build it but there is no detail research. With this project, I could offer a reference to those who want to make this project in the future with detail information.
1.3 Scope of the Project

The scopes of this project can be divided into two. First part is USB controller circuit, this circuit build based on basic data transfer circuits and consist of downstream ports, upstream port, data storage and USB controller IC. Second part of this scope is protection circuits, this circuit aim to make the USB circuit more firm.

Protection circuits consist, 5V voltage regulator, 3V voltage regulator, and power management circuit. Overall project required knowledge in:
- Microcontroller understanding
- Circuits fundamental theory
- PSpice software

1.4 Problem statement

The problem statements of this project are stated as below:

1.4.1 Insufficient Ports

During first year in UMP, I bought a laptop to be using in study. After few month I bought many computer devices as additional features to be add in to my computer such as webcam, printer, mouse, thumb drive and scanner. Because of this, I have insufficient USB ports on my computer to attach new computer devices.
1.4.2 Verification of Doubts

During my seventh semester at UMP, I have taken the course Engineering Projects 1 [BEE4712]. On this course, I’ve being given a choice to choose my topics as my project, one of the topic was to design a USB hub, at that time, I figure out my problem on my first year in UMP, due this problem I insist to take this topic and gaining knowledge in developing my own USB hub.

1.5 Thesis Organization

This thesis consists of five chapters including this chapter. The contents of each chapter are outlined as follows;

Chapter 2 contains a literature review that discussed about the types of elements that will be selected to be used in this project.

Chapter 3 is all about the project methodology. It is specified the method that used in this project in details. It also contains the application and how the testing circuit to be implemented. This will explain how this project is organized and the flow of the process in completing this project.

Chapter 4 presents about the result and discussion that comes from the output of the project. It is presented in phases that are categorized into two phases which are the actual and the simulation results.

The last chapter which is Chapter 5 contains of the conclusion and recommendations to the project. The several difficulties and solution also stated in this chapter.
CHAPTER 2

LITERATURE REVIEW

2.1 USB hubs

Most computers that you buy today come with one or two USB sockets. With so many USB devices on the market today, you easily run out of sockets very quickly. For example, on the computer that I am typing on right now, I have a USB printer, a USB scanner, a USB Webcam and a USB network connection. My computer has only one USB connector on it, so the obvious question is, "How do you hook up all the devices?"[2]

The easy solution to the problem is to buy an inexpensive USB hub. The USB standard supports up to 127 devices and USB hubs are a part of the standard.

Figure 2.1: USB hub in the market.
A hub typically has four new ports, but may have many more. You plug the hub into your computer, and then plug your devices (or other hubs) into the hub. By chaining hubs together, you can build up dozens of available USB ports on a single computer. [2]

Hubs can be powered or unpowered. As you will see on the next page, the USB standard allows for devices to draw their power from their USB connection. Obviously, a high-power device like a printer or scanner will have its own power supply, but low-power devices like mice and digital cameras get their power from the bus in order to simplify them. The power (up to 500 milliamps at 5 volts) comes from the computer. [2]

If you have lots of self-powered devices (like printers and scanners), then your hub does not need to be powered -- none of the devices connecting to the hub needs additional power, so the computer can handle it. If you have lots of unpowered devices like mice and cameras, you probably need a powered hub. The hub has its own transformer and it supplies power to the bus so that the devices do not overload the computer's supply. [2]

### 2.1.1 Power

A bus-powered hub is a hub that draws all its power from the host computer's USB interface. It does not need a separate power connection. However, many devices require more power than this method can provide, and will not work in this type of hub. [3]
In contrast a self-powered hub is one that takes its power from an external power supply unit and can therefore provide full power to every port. Many hubs can operate as either bus powered or self powered hubs. [3]

USB current (related to power) is allocated in units of 100 mA up to a maximum total of 500 mA per port. Therefore a compliant bus powered hub can have no more than four downstream ports and cannot offer more than four 100 mA units of current in total to downstream devices (since one unit is needed for the hub itself). If more units of current are required by a device than can be supplied by the port it is plugged into, the operating system usually reports this to the user. [3]

However, there are many non-compliant hubs on the market which announce themselves to the host as self-powered despite really being bus-powered. Equally there are plenty of non-compliant devices that use more than 100 mA without announcing this fact (or indeed sometimes without identifying themselves as USB devices at all). These hubs and devices do allow more flexibility in the use of power (in particular many devices use far less than 100 mA and many USB ports can supply more than 500 mA before going into overload shut-off) but they are likely to make power problems harder to diagnose.[3]

Some powered hubs do not supply enough power to support a 500mA load on every port. For example, many 7 port hubs come with a 1A power adapter, when in fact seven ports could draw a maximum of 7 x 0.5 = 3.5A, plus power for the hub itself. The assumption is that the user will most likely connect many low power devices and only one or two requiring a full 500mA.[3]
2.1.2 Speed

To allow high-speed devices to operate in their fastest mode all hubs between the devices and the computer must be high-speed. High-speed devices should fall back to full-speed when plugged in to a full-speed hub (or connected to an older full-speed computer port). While high-speed hubs support all device speeds, low and full-speed traffic is combined and segregated from high-speed traffic through a transaction translator. Each transaction translator segregates lower speed traffic into its own pool, essentially creating a virtual full-speed bus. Some designs use a single transaction translator, while other designs have multiple translators. Having multiple translators is only a significant benefit when connecting multiple high-bandwidth full-speed devices. [3]

It is an important consideration that in common language (and often product marketing) USB 2.0 is used as synonymous with high-speed. However, because the USB 2.0 specification, which introduced high-speed, incorporates and supersedes the USB 1.1 specification, any compliant full-speed or low-speed device is still a USB 2.0 device. Thus, not all USB 2.0 hubs operate at high-speed. [3]

2.1.3 Physical layout

A USB network with many devices requires one or more hubs connected to each other. USB hubs can extend a USB network a maximum of five times. The USB specification requires that bus-powered hubs may not be connected in series to other bus-powered hubs. [3]
USB ports on computer housings are usually closely spaced, so that plugging devices into one port may block an adjacent port. This problem is shared by some, but not all, external USB hubs. Star-shaped hubs with each port pointing in a different direction, such as pictured top right, avoid this problem. Aside from practical layouts, novelty USB hubs have also been produced, such as one shaped like the TARDIS, a fictional time-traveling space ship from the BBC science fiction series *Doctor Who*, or another shaped like a nuclear missile launch console complete with a big red button (which shuts down the PC). [3]

Laptop computers may come with many USB ports built in, but a USB hub can consolidate several everyday devices (like a mouse and a printer) into a single port for quick attachment and removal. [3]

Also available are so-called "sharing hubs", which effectively are the reverse of a USB hub, allowing several PCs to access (usually) a single peripheral. They can either be manual (effectively a simple switch-box), or automatic, incorporating a mechanism that recognizes which PC wishes to use the peripheral and switches accordingly. They cannot grant both PCs access at once. Some models, however, have the ability to control multiple peripherals separately (e.g. 2 PCs and 4 peripherals, assigning access separately). Only the simpler switches tend to be automatic, and this feature generally places them at a higher price point too. [3]
2.1.4 Protocol

Each hub has exactly one upstream port and a number of downstream ports. The upstream port connects the hub (directly or through other hubs) to the host. Other hubs or devices can be attached to the downstream ports.

During normal transmission, hubs are essentially transparent: data received from its upstream port is broadcast to all devices attached to its downstream ports; data received from a downstream port is generally forwarded to the upstream port only. This way, what is sent by the host is received by all hubs and devices, and what sent by a device is received by the host but not by the other devices (an exception is resume signaling). [3]

Hubs are not transparent when dealing with changes in the status of downstream ports such as insertion or removal of devices. In particular, if a downstream port of a hub changes status, this change is dealt with an interaction between the host and this hub; the hubs between them act as transparent in this case. [3]

To this aim, each hub has a single interrupt endpoint "1 IN" (endpoint address 1, hub-to-host direction) used to signal changes in the status of the downstream ports. When a device is attached, the hub detects the device pull-up resistor on either D+ or D- and signals the insertion to the host via this interrupt endpoint. When the host polls this interrupt endpoint, it is informed of the presence of the new device. It then instructs the hub (via the default control pipe) to reset the port where the new device is connected. This reset makes the new device assuming address 0, and the host can then interact with it directly; this interaction will result in the assignment of a new (non-zero) address to the device. [3]
2.1.5 Electronics design

Most USB hubs use one or more integrated controller ICs, of which several designs are available from various manufacturers. Most support a four port hub system, but hubs using seven-port hub controllers are also available. Additional features on some hub controllers include control of port LED (sometimes automatic, sometimes under control of the host PC) and PS/2 to USB conversion for mice and keyboards. [3]

2.2 USB hub design

USB hubs can be built that operate in either self-powered or bus-powered mode. Self-powered hubs draw their power from the electrical outlet, while bus-powered hubs draw their power from the USB bus. From the aspect of user experience, hubs operating in self-powered mode have a significant advantage over hubs operating in bus-powered mode for the following reasons:

- A user can plug any bus-powered USB device into any port on a self-powered hub, and the device will always have enough power to function. The power needed by a bus-powered USB device in order to function can range from a few mA up to a maximum of 500mA.
• A user can plug a bus-powered USB device into a port on a bus-powered hub, but the device might not have enough power to function. Specifically:
  o Only low-power bus-powered devices are guaranteed to have enough power available from a bus-powered hub port to operate. Low-power bus-powered devices draw less than 100mA when fully operational. To meet the specification, a bus-powered hub must supply up to 100mA at each port, but must not supply more than 100mA. Typical low-power bus-powered USB devices include mice, keyboards, and other HID devices.
  o A large number of USB devices require between 100mA and 500mA from the hub port when fully operational. These devices will not operate when the user plugs them into a port on a bus-powered hub. Examples of high-power bus-powered devices that will not work with a bus-powered hub include video cameras, page scanners, and floppy disk drives.

Even if it seems unlikely that a user would use USB for a high-power device, the problem with bus-powered USB hub designs is the user's expectation that any USB device can be plugged into any hub and it will work. This is not true with bus-powered hub designs. [4]
2.3 Windows operating system.

High-power, bus-powered devices can draw up to 500mA after they are configured by host system software, but must not draw more than 100mA until they are configured. The device circuitry and self-descriptive information that the host system software requires to enumerate the device are available to the host in this low-power mode. [4]

If the user who is running Microsoft Windows 98 plugs a high-power bus-powered device into a port on a bus-powered hub, the device will not work - and little information will be available to help the user understand why. Windows 98 shows the device as disabled in Device Manager, but does not warn the user with a message, for example, at the time of the hot-plug event. [4]

Windows 98 Second Edition, Windows Me, Windows 2000, and Windows XP give the user more help in such a situation. The device is shown as disabled in Device Manager and the user is also presented with a message that indicates why the device is not operating. However, it is still up to the user to locate an unused port on the PC platform or on a self-powered hub connected to the PC platform. Then the user must disconnect the high-power bus-powered device from the bus-powered hub and reconnect it to a port that will supply enough power. [4]

2.4 .HEX calculation

Intel 8-bit Hex File Format is the most common hex file format used in the world as far as I know. There is also Motorola Hex file format and maybe other. Creating applications with AVR-GCC we usually select ihex output file format what means Intel hex file format. [5]
1 Record Marker: The first character of the line is always a colon (ASCII 0x3A) to identify the line as an Intel Hex file

2 – 3 Record Length: This field contains the number of data bytes in the register represented as a 2-digit hexadecimal number. This is neither the total number of data bytes, not including the checksum byte nor the first 9 character of line.

4 – 7 Address: This field contains the address where the data should be loaded into the chip. This is a value from 0 to 65,535 represented as a 4-digit hexadecimal value.

8 – 9 Record Type: This field indicates the type of record for this line. The possible values are: 00=Register contains normal data, 01=End of file; it is usually “:0000001FF”, 02=Extended address.

10 - ? Data Bytes: The following bytes are the actual data that will be burned into the EEPROM. The data is representing as 2-digit hexadecimal values.

Last 2 Character Checksum: The last two characters of the line are checksum for the line. The checksum value is calculated by taking the two’s compliment of sum of all the preceding data bytes, excluding the checksum byte itself and the colon at the beginning of the line.

Table 2.1: Description of .HEX bytes
Check sum calculation example:

:040F4009F4F089522

Taking all the data bytes above, we have to calculate the checksum based on the following hexadecimal values:

04+0F+40+00+9F+4F+08+95 = 1DE

The value is greater then we leave part which is less than FF,so we get DE.

Then Twos complement is 100h-DE=22

In Intel Hex File Format there are six types of record types:

- 00 – data record;
- 01 – End of file record. Usually it is “:00000001FF”;
- 02 – Extended Segment address record. This indicates segment base address when 16 bits is not enough for addressing memory;
- 03 – Start segment address record. Indicates initial segment base address.
- 04 – Extended Linear Address Record – allows 32 bit addressing.
- 05 – Start Linear Address Record.
CHAPTER 3

METHODOLOGY

3.1 Project Overview

This project only involves hardware. The hardware can be divided into two parts which is USB controller and USB protection circuit. USB controller circuit can be divided into four stages which are the downstream ports, upstream ports, USB controller IC and SPI serial EEPROM. USB protection circuit can be divided into three stages which are 5V voltage regulator, 3V voltage regulator and power management circuit. When all the circuit is functioning and have the desired output, the circuits will be integrated into one massive functioning circuit.

3.2 Hardware explanations

3.2.1 USB controller circuit

USB controller circuits consist of four stages which are downstream ports, upstream port, SPI serial EEPROM and USB controller IC.
3.2.1.1 Downstream port circuit

![Figure 3.1: Basic downstream circuit](image)

Downstream port is the place to plugging in computer device plug into the hub. Downstream circuit shown in figure 3.1 above describe how downstream port be attaching into hub. Downstream D+ and D– pull-down resistors are incorporated in USB controller for each port. Prior to the hub being configured, the ports are driven SE0 (Single Ended Zero, where both D+ and D– are driven LOW) and are set to the unpowered state. Once the hub is configured, the ports are not driven, and the host may power the ports by sending a “SetPortPower” command to each port. After a port is powered, any connect or disconnect event is detected by the hub. Any change in the port state is reported by the hub back to the host through the Status Change Endpoint (endpoint 1). Upon receipt of “SetPortReset” command from the host, the hub will

- Drive SE0 on the corresponding port
- Put the port in an enabled state
- Enable the green port indicator for that port
- Enable babble detection once the port is enabled.
Babble consists of either unterminated traffic from a downstream port (or loss of activity), or a non-idle condition on the port after EOF2. If babble is detected on an enabled port, that port will be disabled. A ClearPortEnable command from the host will also disable the specified port. Downstream ports can be individually suspended by the host with the SetPortSuspend command.

If the hub is not suspended, any resume will be confined to that individual port and reflected to the host through a port change indication in the Hub Status Change Endpoint. If the hub is suspended, a resume on this port will be forwarded to the host, but other resume events will not be seen on that port. The host may resume the port by sending a ClearPortSuspend command.
3.2.1.2 Upstream port circuit

![Circuit Diagram]

**Figure 3.2**: Basic upstream circuit

The upstream port includes the transmitter and the receiver state machine. The Transmitter and Receiver operate in high-speed and full-speed depending on the current hub configuration. The transmitter state machine monitors the upstream facing port while the Hub Repeater has connectivity in the upstream direction.

This monitoring activity prevents propagation of erroneous indications in the upstream direction. In particular, this machine prevents babble and disconnects events on the downstream facing ports of this hub from propagating and causing the hub to be disabled or disconnected by the hub to which it is attached. This allows the Hub to only disconnect the offensive port on detecting babble from it.
3.2.1.3 SPI serial EEPROM data storage

Figure 3.3: Basic PSI serial EEPROM circuit

Systems using USB controller IC must have an external EEPROM in order for the device to have a unique VID, PID, and DID. The USB controller IC can talk to SPI EEPROM that are double byte addressable only. USB controller IC uses the command format from the '040 parts. The USB controller IC cannot talk to '080 EEPROM parts, as the read command format used for talking to ‘080 is not the same as ‘040. The '010s and '020s uses the same command format as used to interface with the ‘040 and hence these can also be used to interface with the USB controller IC.

When used in default mode, only a unique VID, PID, and DID must be present in the external SPI EEPROM. The contents of the EEPROM must contain this information in the following format:
### Table 3.1: Data storage default Byte configuration

<table>
<thead>
<tr>
<th>Byte</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>3</td>
<td>00</td>
</tr>
<tr>
<td>4</td>
<td>00</td>
</tr>
<tr>
<td>5</td>
<td>00</td>
</tr>
<tr>
<td>6</td>
<td>00</td>
</tr>
</tbody>
</table>

### Table 3.2: Data storage configured Byte configuration

<table>
<thead>
<tr>
<th>Byte</th>
<th>Value (MSB-&gt;LSB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D2</td>
</tr>
<tr>
<td>1</td>
<td>B4</td>
</tr>
<tr>
<td>2</td>
<td>04</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>65</td>
</tr>
<tr>
<td>5</td>
<td>07</td>
</tr>
<tr>
<td>6</td>
<td>00</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
</tr>
<tr>
<td>8</td>
<td>FF</td>
</tr>
<tr>
<td>9</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
</tr>
<tr>
<td>11</td>
<td>32</td>
</tr>
<tr>
<td>12</td>
<td>81</td>
</tr>
</tbody>
</table>
**Bytes descriptions**

**Byte 0: 0xD2**
Needs to be programmed with 0xD2

**Byte 1: VID (LSB)**
Least Significant Byte of Vendor ID

**Byte 2: VID (MSB)**
Most Significant Byte of Vendor ID

**Byte 3: PID (LSB)**
Least Significant Byte of Product ID

**Byte 4: PID (MSB)**
Most Significant Byte of Product ID

**Byte 5: DID (LSB)**
Least Significant Byte of Device ID

**Byte 6: DID (MSB)**
Most Significant Byte of Device ID

**Byte 7: EnableOvercurrentTimer[3:0], DisabledOvercurrentTimer[3:0]**
Count time in ms for filtering overcurrent detection. Bits 7–4 are for an enabled port, and bits 3–0 are for a disabled port. Both range from 0 ms to 15 ms.

**Byte 8: ActivePorts[3:0], RemovablePorts[3:0]**
Bits 7–4 are the ActivePorts[3:0] bits that indicates if the corresponding port is usable. For example, a two-port hub that uses ports 1 and 4 would set this field to 0x09. The total number of ports reported in the Hub Descriptor: bNbrPorts field is calculated from this. Bits 3–0 are the Removable-Ports [3:0] bit that indicates whether the corresponding port is removable (set to HIGH). This bit’s values are recorded appropriately in the HubDescriptor:DeviceRemovable field. Default: 0xFF.

**Byte 9: MaximumPower**
MaxPower field and is the current in 2-mA intervals that is required from the upstream hub.
Byte 10: HubControllerPower
HubContrCurrent field and is the current in milliamperes required by the hub controller. Default: 0x64 = 100 mA.

Byte 11: PowerOnTimer
PwrOn2PwrGood field and is the time in 2-ms intervals from the SetPortPower command until the power on the corresponding downstream port is good.

Byte 12: IllegalHubDescriptor, Unused, FullspeedOnly, NoPortIndicators, Reserved, GangPowered, SingleT-TOnly, NoEOPatEOF1

- **Bit 7**: IllegalHubDescriptor: For GetHubDescriptor request, some USB hosts use a DescriptorType of 0x00 instead of HUB_DESCRIPTOR, 0x29. According to the USB 2.0 standard, a hub must treat this as a Request Error, and STALL the transaction accordingly (USB 2.0, 11.24.2.5). For systems that do not accept this, the IllegalHubDescriptor configuration bit may be set to allow TetraHub to accept a DescriptorType of 0x00 for this command. Recommended setting is 1.
- **Bit 6**: Unused: This bit is an unused, don’t care bit and can be set to anything.
- **Bit 5**: Fullspeed: Only configures the hub to be a full-speed only device. Default set to 0.
- **Bit 4**: NoPortIndicators: Turns off the port indicators and does not report them as present in the HubDescriptor, wHubCharacteristics b7 field. Default set to 0.
- **Bit 3**: Reserved: This bit is reserved and should not be set to 1. Must be set to 0.
- **Bit 2**: GangPowered: Indicates whether the port power switching is ganged (set to 1) or per-port (set to 0). This is reported in the HubDescriptor, wHubCharacteristics field, b4, b3, b1, and b0. Default set to 0.
- **Bit 1**: SingleTTOnly: Indicates that the hub should only support single Transaction Translator mode. This changes various descriptor values. Default set to 0.