CIRQUE®

Composite USB

Application Note

Revision 1.0

JUNE 2012
GP-AN-120515

Reference hardware and sample firmware are available upon request.

Revision History

Date	Previous Revision	Current Revision	Description
JUNE 2012		1.0	Documentation creation.

Copyright and Patent Information

GlidePoint and GlidePoint Technology are protected under U.S. Patents # 5,565,658; 5,757,368; 5,861,875; 6,222,528; 5,767,457; 6,473,069; 6,730,863; 6,262,717; 6,639,586; 7,151,528; 7,808,487; 7,394,453; 7,306,144; 7,564,449; 7,817,145 and other U.S. and International patents pending.

Table of Contents

1. Introduction

This document describes how to interface a Cirque Touch Module and a keyboard into one USB product. Included in this document:

- Overview of USB Composite devices
- How operating systems display composite devices
- How the Cirque driver attaches to composite devices
- Firmware considerations

This document assumes the reader has some prior knowledge of USB, C, C++, and/or C# programming and computer operating systems.

Note: *The information in this document is for a 64-bit Windows 7 machine. Other versions will be documented in future releases.*

2. Overview of a Composite Device

Designing a product that includes two (2) or more USB devices requires either a USB hub or firmware defining a Composite device. USB hubs add cost to the product, so many vendors prefer to design a Composite device.

2.1 USB Composite Devices

A composite device is a USB device that has more than one interface and each interface is controlled by a different device driver running on the host (or by a different instance of the same driver running on the host). For example:

- An audio-visual device can have two interfaces, audio (an Audio class interface) and video (an Imaging class interface).
- A telephony device can have three interfaces: audio (Audio class interface), a keypad (a HID class interface), and a modem (a Communications class interface). The host can use interface power management to put the modem in a low-power wake-enabled mode when only the keypad and audio interfaces are being used.

3. Displaying Composite Devices in Operating Systems

Operating systems that support USB treat a composite device as two different USB devices. Designers are required to define all the devices of the composite device in the firmware. When a USB Composite Device enumerates, the OS generates hardware IDs for the parent and children based on the VID, PID, and REV values reported by the firmware.

Note: *Connecting USB devices to an internal hub does not require each device to be defined in one firmware code.*

- Windows 7 generates a primary hardware ID for the parent (composite) device as USB\VID_xxxx&PID_xxxx&REV_xxxx
- Installs the USB common class generic parent driver on it, which creates a child device for each interface exposed by the firmware.
- The primary hardware IDs for child devices have the format USB\VID_xxxx&PID_xxxx&REV_xxxx&MI_xx
- The first child is defined in the firmware as MI _00, the second is MI_01, and so forth.
- Child devices are identified in the Device Manager   “USB Input Device,” unless a custom driver gives them another name.
- Windows 7 also generates secondary hardware IDs for the parent and children, which are similar to the primary IDs, but contain no REV number.
- Unless the vendor provides a special custom driver, Windows 7 will install a generic HID class driver on each child device, which will enumerate grandchildren based on the HID report descriptor defined in the firmware of the child device.
  - One grandchild will be created for each top-level collection (TLC) in the report descriptor and assigned a hardware ID in the format

    HID\VID_xxxx&PID_xxxx&REV_xxxx&MI_xx
  - if the report descriptor has only one TLC or HID\VID_xxxx&PID_xxxx&REV_xxxx&MI_xx&Colxx
  - If the report descriptor has more than one TLC, where Col01 will correspond to the first TLC, Col02 to the second, and so on.
- The grandchildren usually represent common devices known as Human Interface Devices or HID devices. They either require no drivers or are serviced by the appropriate class driver, such as a keyboard or mouse driver, which is provided by the OS.

An example of how the Device Manager in Windows® displays a Composite device is shown in Table 1 on page 6.

Table 1. Composite Devices Create the following in the Windows Device Manager

Device Name	Type	Hardware IDS
USB Composite Device	Universal Serial Bus Controller	USB\VID_0461&PID_0055&REV_0100 USB\VID_0461&PID_0055
USB Input Device	Human Interface Devices	USB\VID_0461&PID_0055&REV_0100&MI_00 USB\VID_0461&PID_0055&MI_00
HID-compliant mouse	Mice and other pointing devices	USB\VID_0461&PID_0055&REV_0100&MI_00 USB\VID_0461&PID_0055&MI_00 HID_DEVICE_SYSTEM_MOUSE HID_DEVICE_UP:0001_U:0002 HID_DEVICE
USB Input Device	Human Interface Devices	USB\VID_0461&PID_0055&REV_0100&MI_01 USB\VID_0461&PID_0055&MI_01
HID Keyboard Device	Keyboards	USB\VID_0461&PID_0055&REV_0100&MI_01 USB\VID_0461&PID_0055&MI_01 HID_DEVICE_SYSTEM_KEYBOARD HID_DEVICE_UP:0001_U:0006 HID_DEVICE

4. How the Cirque Driver Attaches to Composite Devices

This section provides information about how the Cirque driver will attach to a composite device.

- The Cirque driver does not attach to the parent composite device.
- The Cirque driver will attach to a child device if it finds a match for the hardware ID in its .INF file.
- The children may use a custom driver provided by the vendor or a generic driver provided by the operating system.
- Custom drivers use an .INF file as a database to find and match Vendor ID numbers (VID), Product ID numbers (PID), and in the case of a composite device, the child number (&MI_01).
- Not every custom device requires its own INF file. Many devices that use the system's class drivers can use the INF file that Windows provides for the class.
- Windows XP and later Operating Systems prefer signed drivers. Cirque drivers are signed and have passed WHQL testing. Developing a signed driver can be costly and time consuming. For this reason, Cirque offers a partnership to selected vendors to include the vendors VID and PID in Cirque's INF file.
- Vendor ID, Product ID, Firmware revision, Manufacturer string, Product string, and Serial Number string are defined in the USB Device Descriptor.

4.1 Firmware Considerations

Many vendors of microcontrollers that have USB capability offer sample USB composite firmware code. They also offer a wide knowledge base of information.

Most operating systems have HID drivers installed for both the mouse and the keyboard.

A mouse and keyboard combo USB device may be implemented by using multiple interfaces. Each interface uses one or more endpoints; both share endpoint 0 as a control port. The mouse and keyboard information is described in the Interface Descriptors located in the firmware. One Interface descriptor is used for the mouse, and another for the keyboard. No Report IDs are needed for this method and a simple mouse data protocol is:

- Mouse_data [0]=button state
- Mouse_data [1]=x delta
- Mouse_data [2]=y delta
- Mouse_data [3]=wheel delta

Another USB mouse and keyboard combination device may be implemented by using multiple HID reports. One HID report is used for the mouse; another HID report is used for the keyboard. Each report must include a Report ID to designate which report to use when data is sent to the USB HID drivers. When a Report ID is used, it must be the first byte sent to the Host.

When a Report ID number is added to the HID report descriptor, the first byte of data sent to the Host must be the Report ID. For example, a simple mouse data protocol sent to the Host is:

- Mouse_data[0]=HID Report ID
- Mouse_data [1]=button state
- Mouse_data [2]=x delta
- Mouse_data [3]=y delta
- Mouse_data [4]=wheel delta

This method of multiple HID reports is easy to implement, but you lose bandwidth as the mouse and keyboard must share the same endpoint.

5. Cirque's Module Hardware, Firmware, and Driver Overview

When the USB cable of a Touch Module connects to a PC, the USB Host takes control and enumerates the module as a HID mouse device.

The Touch Module powers-up in Relative mode as a USB HID mouse device. The Pinnacle X, Y, Z data registers of the Touch Module contains relative mouse data. Relative data represents a current mouse cursor position relative to the last position. The relative data range for both the X and Y-axis is +127 to -128. Normally this data is +1 or -1 in either axis.

If a Cirque driver has been previously installed, the driver will recognize the VID as a Cirque Touch Module and take control of the device.

The firmware must allow the device to enumerate as a relative-data mouse, then, if instructed by the driver, switch to either IntelliMouse mode or Absolute mode (Table 2).

When the Cirque driver attaches to the Touch Module after enumeration, the driver will send a value of 0x64 to the Touch Module to set the module into Absolute mode. The firmware receives this value and sets bit 2 in the Pinnacle register FeedConfig1 address 0x04.

A Read-Modify-Write for this register is recommended

- Read Pinnacle register 0x04
- OR the received byte with 0x02
- Write the modified byte back to the Pinnacle register 0x04

Absolute data reports X, Y positions and Z (finger signal strength) data to the system. It also reports the optional Px data that is the foundation for creating multi-finger Area Gestures.

The absolute data mode allows the designer a wider range of freedom to compute the data and design the desired touch functions.

The Touch Module will automatically change the data contained in the X, Y, Z registers to the Absolute Data Protocol.

Firmware sets bit 2 of the FeedConfig1 register address 0x04 to enter Absolute mode.

Absolute data range:

- Pinnacle 2.2 range: X= 0 - 2047, Y= 0 - 1535
- Pinnacle AG range: X= 0 - 1919, Y= 0 - 1407

Table 2. Mouse, IntelliMouse and Cirque Driver data - all modes

Pinnacle Register Name	AG Address (Dec)	AG Address (Hex)	P2.2 Address (Dec)	P2.2 Address (Hex)	Description/Contents
Numb-Fingers	15	0F	-	-	Number of fingers on trackpad in each axis
PacketByte_0	16	10	18	12	(X-Y Sign/Button (or tap) REL mode) (NF0-1 and button ABS mode)
PacketByte_1	17	11	19	13	X-Position Delta Low Byte REL and ABS modes
PacketByte_2	18	12	20	14	Y-Position Delta Low Byte REL and ABS modes

Table 2. Mouse, IntelliMouse and Cirque Driver data - all modes

Pinnacle Register Name	AG Address (Dec)	AG Address (Hex)	P2.2 Address (Dec)	P2.2 Address (Hex)	Description/Contents
PacketByte_3	19	13	21	15	X-Y Pos High Byte/Wheel Count REL mode)
PacketByte_4	20	14	22	16	(X-Y high byte ABS mode) (Z-Level REL mode) (Z-low Y-high ABS mode)
PacketByte_5	21	15	23	17	X-Pos High Byte ABS
PacketByte_6 (AG only)	22	16	-	-	X-Y8 Pos High Byte ABS
PacketByte_7 (AG only)	23	17	-	-	Y-Pos Low Byte ABS

Data in the Pinnacle registers change, depending on the current mode: Relative, IntelliMouse, or Absolute (Table 3).

Table 3. Relative Data Packet Format From the Pinnacle and As Needed By the HID Driver

Register	Byte #	Bit	Description	AG Address	P2p2
PACKETBYTE_0	0	0	1=Left button (tap) pressed, 0=Left button (tap) released	0x10	0x12
		1	1=Right button (tap) pressed, 0=Right button (tap) released	0x10	0x12
		2	1=Middle button pressed, 0=Middle button released	0x10	0x12
		3	Always 1	0x10	0x12
		4	X data sign (1=negative)	0x10	0x12
		5	Y data sign (1=negative)	0x10	0x12
		6	0	0x10	0x12
		7	0	0x10	0x12
PACKETBYTE_1	1	0-7	X count data (bit0=LSB)	0x11	0x13
PACKETBYTE_2	2	0-7	Y count data (bit0=LSB)	0x12	0x14

The IntelliMouse Packet Format From the Pinnacle is shown in Table 4.

Table 4. IntelliMouse Packet Format From the Pinnacle and As Needed By the HID driver

Register	Byte #	Bit	Description	AG Address	P2p2
PACKETBYTE_0	0	0	1=Left button (tap) pressed, 0=Left button (tap) released	0x10	0x12
		1	1=Right button (tap) pressed, 0=Right button (tap) released	0x10	0x12
		2	1=Middle button pressed, 0=Middle button released	0x10	0x12
		3	Always 1	0x10	0x12
		4	X data sign (1=negative)	0x10	0x12
		5	Y data sign (1=negative)	0x10	0x12

Table 4. IntelliMouse Packet Format From the Pinnacle and As Needed By the HID driver

Register	Byte #	Bit	Description	AG Address	P2p2
		6	0	0x10	0x12
		7	0	0x10	0x12
PACKETBYTE_1	1	0-7	X count data (bit0=LSB)	0x11	0x13
PACKETBYTE_2	2	0-7	Y count data (bit0=LSB)	0x12	0x14
PACKETBYTE_3	3	0-7	Wheel count (bit0=LSB)	0x13	0x15

The Absolute Packet Format From the Pinnacle 2.2 (SPI or I2C) is shown in Table 5.

Table 5. Absolute Packet Format From the Pinnacle 2.2 (SPI or I2C)

Register	B7	B6	B5	B4	B3	B2	B1	B0	P2p2 Address
PACKETBYTE_0	0	0	SW5	SW4	SW3	SW2	SW1	SW0	0x12
PACKETBYTE_1	0	0	Ps	P4	P3	P2	P1	P0	0x13
PACKETBYTE_2	X7	X6	X5	X4	X3	X2	X1	X0	0x14
PACKETBYTE_3	Y7	Y7	Y5	Y4	Y3	Y2	Y1	Y0	0x15
PACKETBYTE_4	Y11	Y10	Y9	Y8	X11	X10	X9	X8	0x16
PACKETBYTE_5	Touch	0	Z5	Z4	Z3	Z2	Z1	Z0	0x17

Notes:

- SW0-5 = status of buttons 0-5
- PS = the palm detection data
- X0-X10 = Absolute X coordinates (X10=MSB) Range 0-2047
- Y0-Y10 = Absolute Y coordinates (Y10=MSB) Range 0-1535
- Z0-Z5 = Z (finger signal strength) value (Z5=MSB)
- Touch flag (1=touch)

The Absolute Packet Format From the Pinnacle AG1.4 (SPI or I2C) is shown in Table 6.

Table 6. Absolute Packet Format From the Pinnacle AG1.4 (SPI or I2C)

Register	B7	B6	B5	B4	B3	B2	B1	B0	AG Address
PACKETBYTE_0	NF1	NF0	SW5	SW4	SW3	SW2	SW1	SW0	0x10
PACKETBYTE_1	X7	X6	X5	X4	X3	X2	X1	X0	0x11
PACKETBYTE_2	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	0x12
PACKETBYTE_3	Y11	Y10	Y9	Y8	X11	X10	X9	X8	0x13
PACKETBYTE_4	PY10	PY9	Z5	Z4	Z3	Z2	Z1	Z0	0x14
PACKETBYTE_5	PX7	PX6	PX5	PX4	PX3	PX2	PX1	PX0	0x15
PACKETBYTE_6	PY8	PX14	PX13	PX12	PX11	PX10	PX9	PX8	0x16
PACKETBYTE_7	PY7	PY6	PY5	PY4	PY3	PY2	PY1	PY0	0x17

Notes: Absolute Packet Format From the Pinnacle AG1.4 (SPI or I2C) Abbreviations:

- NF1, NF0 = number of fingers on the pad
- SW0-5 = status of buttons 0-5
- PY0-PY0 = Absolute Gesture data on the Y-axis
- PX0-PX14 = Absolute Gesture data on the X-axis
- X0-X10 = Absolute data for X coordinates (X10=MSB) Range 0-2047
- Y0-Y10 = Absolute data for X coordinates (Y10=MSB) Range 0-1535
- Z0-Z5 = Z (finger signal strength) value (Z5=MSB)

In Relative and IntelliMouse modes, in both Pinnacle AG and Pinnacle 2.2 devices, the registers may be read and sent to the USB Host without additional formatting.

In Absolute mode, the Cirque driver is expecting a USB packet in a different format than the data in the Pinnacle registers.

When using the Pinnacle AG, the Pinnacle register Number-of-Fingers must also be read to properly format USB packets for the Cirque driver.

Table 7. IntelliMouse Packet Format From the Pinnacle and As Needed By the HID driver

Register Name	Description	AG Address	P2p2 Address
Number of Fingers	Number of fingers on TM	0x0F	N/A

For Pinnacle AG, the USB packets must be formatted as described below:

Table 8. Pinnacle Packet Format Needed By the Cirque Driver

Byte 12 LSB	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
0000 0 LFB NF1 NF0	0000 0 PY8-PY10	PY0-PY7	0 PX8-PX14 0PPP PPPP	PX0-PX7 PPPP PPPP	0ZZZ ZZZZ	00YY YYYY	YYYY YXXX	XXXX XXXX	wwww wwww	yyyy yyyy	xxxx xxxx	0000 0bbb

Notes: Pinnacle Packet Format Abbreviations:

- LFB = Large Finger Bit (tracking)
- NF1, NF0 = Number of fingers
- PY0-PY10 = Palm data for Y axis
- PX0-PX14 = Absolute X data
- Z = finger signal strength data
- Y = Y axis Absolute data
- X = X axis Absolute data
- w = wheel data for Intellimouse
- y = Y axis Relative data (normal mouse mode)
- x = X axis Relative data (normal mouse mode)
- b = button data (3 buttons)

6. How the Firmware Reads the Pinnacle Registers and Formats to the Absolute Mode

The following code is an example of firmware reading the Pinnacle registers and formatting to Absolute mode:

```
PinnacleIO_ReadRegister(Address=0x0F,&PinnRegs[8]);  // number of fingers
PinnacleIO_ReadMemory(Address=0x10, # values to read=4, &PinnRegs[4]);

USBPacket[0] = PinnRegs[0] & 0x07;  //buttons
USBPacket[4] = PinnRegs[1];    //first 8 bits of X absolute
USBPacket[5] = ((PinnRegs[3] & 0x07) | ((PinnRegs[2]<<3)&0xF8));
    //last 3 bits of X absolute + first 5 bits of y absolute
USBPacket[6] = (((PinnRegs[2] & 0xE0)>>5)&0x07) | ((PinnRegs[3]>>1) & 0x38);
                         //last 6 bits of Y
USBPacket[7] = PinnRegs[4] & 0x3F;
USBPacket[8] = PinnRegs[5];
USBPacket[9] = PinnRegs[6] & 0x7F;
USBPacket[10] = PinnRegs[7];
USBPacket[11] = ((((PinnRegs[6]>>7)&0x01) | ((PinnRegs[4]>>5)&0x06)) & 0x07);
USBPacket[12] = ((PinnRegs[0]>>6)&0x03) | ((PinnRegs[8]>>5)&0x04);
```

6.1 Firmware Interface To the Cirque Driver

- The firmware interfaces to the Cirque driver through USB.
- When the Cirque driver attaches to the Touch Module after enumeration, the driver will send a value of 0x64 to the Touch Module.
- The FW must set bit 1 (0-7 bits) in the register FeedConfig1 register address 0x04
- A Read-Modify-Write for this register is recommended
    - Read Pinnacle register 0x04
    - OR the byte with 0x02 (bit 1)
    - Write Pinnacle register 0x04

7. USB Setup Packet Format for Cirque Vendor Commands

This section describes the format of the USB Setup packet and commands.

Table 9. Vendor Commands

bmRequestType	bRequest	wValue	wIndex	wLength
Request Type	Command ID	Memory Mask	Memory Address	Always 000h

bmRequestType: This value determines the packed Request Type and must be 0C1h, 040h, or 041h for the packet to be processed as a Cirque vendor command.

bRequest: This value determines which Cirque vendor command to execute. It must be one of the following values:

Table 10. bRequest Values

Value	Command	Comments
0x64	Set Extended Mode	Enables the Gestures
0x65	Clear Extended Mode	Disable the Gestures, returns the device to a HID mouse
0x66	Write Memory	Writes data to specific registers
0x67	AND Memory	Used in conjunction with the Read and Write commands
0x68	OR Memory	Used in conjunction with the Read and Write commands
0x69	Read Memory	Reads data from specific registers
0x6A	Get Status	Used extensively during initialization

wValue: This sixteen bit value contains the Memory Mask to be used when executing a Write/AND/OR Memory Location command. The lower twelve bits contain the mask, the upper four bits are ignored.

wIndex: This sixteen bit value contains the Memory Address to be used when executing a Write/AND/OR/Read Memory Location command. The lower twelve bits contain the mask, the upper four bits are ignored.

wLength: This value is not used by any Cirque vendor command. It should always be 000h.

7.1 Cirque Vendor Commands

06Ah-Get Status: This command causes the firmware to return the current eight-byte status. This command may be issued at any time. The eight bytes are defined in table.

Table 11. Get Status Byte

Byte 0	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
Device State Flags	Standard Mode Flag	Standard +Wheel Mode Flag	Extended Mode Flag	Busy Flag	Power Cycle Flag	Unused

Bits 0-2 are used to indicate the current asic mode and the type of data being reported in Endpoint1 packets. The following table shows the valid modes.

Table 12. Absolute Packet Format From the Pinnacle 2.2 (SPI or I2C)

Bits Reported Set	Device Mode
0 only	Device is reporting standard data.
1 only	Device is reporting standard+wheel data.
2 and 0	Device is reporting extended mode data and will return to standard mode if an Exit Extended Mode Command is issued.
2 and 1	Device is reporting extended mode data and will return to standard+wheel mode if an Exit Extended Mode Command is issued.

Bit 3 is used to indicate whether or not the firmware is ready to accept Cirque vendor commands other that Get Status. If this bit is set, the firmware is not ready to accept commands or is busy processing a Cirque vendor command. Issuing Cirque vendor commands while this bit is set results in a stall.

Bit 4 is used to notify the caller that the firmware was forced to reset the Cirque asic and that any memory modifications to the asic may be lost. This bit is cleared after each Get Status call.

7.1.0.1Bytes

Byte 1: USB Firmware Version

This byte contains the USB firmware version with the major version in the high nibble and the minor version in the low nibble (23 = 2.3)

Byte 2: Cirque Asic Firmware Version

This byte contains the Cirque asic firmware version with the major version in the high nibble and the minor version in the low nibble (55 = 5.5)

Byte 3: Cirque Asic HCOs 0-7 This byte contains the values of HCOs 0-7 in bits 0-7 respectively. The following table describes HCOs 0-7.

Table 13. HCOs 0-7

HCO	Description
0	180 Rotate.
1	Unused
2	PID 0
3	PID 1
4	PID 2
5	Extended Default
6	Unused
7	Flip-X

Byte 4: Cirque Asic Memory Value Low Byte

Byte 5: Cirque Asic Memory Value High Byte

These bytes are only valid in the first Get Status call with no busy flag after a successful Read Memory Location call. If valid, they will contain the value at the location used in the Read Memory Location call.

Byte 6: Cirque Vendor Command Return Value

This byte is only valid in the first Get Status call with no busy flag after a Cirque vendor command call. It contains the return value of the Cirque vendor command. Any value other than zero indicates failure.

Byte 7: Cirque Asic HCOs 8 - 11

This byte contains the values of HCOs 8-11 in bits 0-4 respectively. The following table describes HCOs 8-11.

Note: *HCOs are determined by reading the resistors populated near the ASIC. If you prefer, you may contact Cirque for the ASIC Firmware Version for the ASIC your product will be using. You can then use this as a constant in your FW code.*

Table 14. HCOs 8-11

HCO	Description
8	OEM
9	PID 3
10	No Right Tap
11	Wheel Mode

064h - Set Extended Mode: This command causes the firmware to put the Cirque asic into extended mode and begin reporting extended mode data in the Endpoint1 data packets.

065h - Exit Extended Mode: This command causes the firmware to take the Cirque asic out of extended mode and begin reporting standard or standard+wheel data in Endpoint1 packets. The mode returned to (standard or standard+wheel) depends on the mode the unit was in before it was put into extended mode.

066h - Write Memory Location: This command causes the firmware to write to the Cirque asic the value contained in Memory Mask (wValue) to the location contained in Memory Address (wIndex).

067h - AND Memory Location: This command causes the firmware to AND the Cirque asic location contained in Memory Address (wIndex) with the value contained in Memory Mask (wValue).

068h - OR Memory Location: This command causes the firmware to OR the Cirque asic location contained in Memory Address (wIndex) with the value contained in Memory Mask (wValue).

069h - Read Memory Location: This command causes the firmware to read the Cirque asic location contained in Memory Address (wIndex). The value contained in that location can be read by examining bytes four and five of the first Get Status call with no busy flag after the Read Memory Location call.

8. Basic Vendor Command Sequence

The Basic Vendor Command sequence (except Get Status) is shown below in Figure 1.

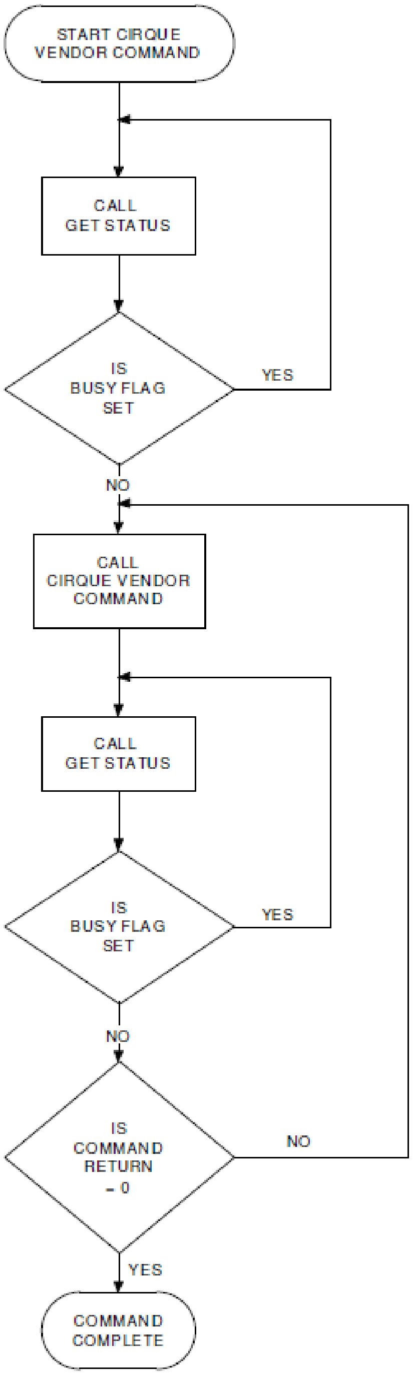


Figure 1. Basic Vendor Command sequence

9. Partners with Cirque

Cirque would add the partner's Vendor ID number (VID) and Product ID number (PID) to Cirque's driver.inf file. When the device connects to a USB Host, the device would enumerate as a composite device with a parent and 1 or more child numbers. USB assigns each child member an extension. Normally, these extensions are &MI_00 for the first child device, &MI_01 for the second.

Example: VID_0333&PID_0222&REV_1111&MI_00

The Cirque driver would attach to the child Touch Module. The touch module should have a single top-level collection, which should be a mouse collection reporting mouse data packed as described above.

The partner's driver may attach to the keyboard. The partner may decide to allow the HID driver to control the keyboard.

Please contact Cirque for more information (See Contact Information on page 30).

10. Sample Code

```
In main.c:
void main(void)
{
InitializeHW();// set up timing, USB, SPI, etc.
Detach_usb();// causes the USB to restart / enumerate
while (TRUE)// forever loop
{
usb_task();// if you are polling for USB activity then check now
// otherwise, clear ISR flag and re-enable ISR

if  (USB_is_enumerated())
{
Check_USB_mouse();
Check_USB_keyboard();
delay(10);// delay is in ms
}
}
}
///////////////////////////////////////////////////////////////////////////////
//
// Check_USB_mouse()
//
// Sends a packet of data containing mouse information.
// USB specifications define the data to send to the HID driver:
//   usb_out_mouse_data[0]=button state
//   usb_out_mouse_data[1]=x delta
//   usb_out_mouse_data[2]=y delta
//   usb_out_mouse_data[3]=wheel delta
//
///////////////////////////////////////////////////////////////////////////////
void Check_USB_mouse(void)
{
   int8 usb_out_mouse_data[4];
  usb_out_mouse_data[0] = 0;   // no buttons pressed  0x01 = button 1 pressed
 usb_out_mouse_data[1] = X_data;
 usb_out_mouse_data[2] = Y_data;
 usb_out_mouse_data[3] = 0;// no wheel or scroll data to send

/* if the USB TX buffer is not full then send the 4 packets starting at packet[0] */
/* bool  Send_USB_packet(end point, pointer to buffer, length, toggle )*/
if (Send_USB_packet(1,&usb_out_mouse_data,4,TOGGLE_USB)) ;
}
///////////////////////////////////////////////////////////////////////////////
//
// Check_USB_keyboard()
//
// Sends keyboard data
// USB specifications define the data to send to the HID driver:
//     kb_tx_key[0] = modifier (an 8-bit bitmap of shift, tab, alt keypress)
//     kb_tx_key[1] = const 0
//     kb_tx_key[2:6] = an array of held down keys.  a=4, b=5, etc.
//
//  if msg[2:7]={0} then no keys are held down
//
//     rx_msg[0] = 5bit bitmap of led status
//
// for this example, a button on the dev board was pressed to send a ‘b’ character
//    if kb_tx_key[0] show that the shift key is pressed, the the ‘B’ char is sent
//    to a monitor
//    A text editor should be open to receive and display the character
//
```

```
//    Refer to the USB HID tables - LED section if necessary
//////////////////////////////////////////////////////////////////////////////
void Check_usb_keyboard(void)
{
static char kb_tx_key[7] = {0,0,0,0,0,0,0};
 static char LEDs;

// press a button to send a “b” char to the monitor
// the button is pulled up until pressed to gnd

if (DEV_BUTTON==0)  kb_tx_key[2]=5;// b character
   else kb_tx_key[2]=0;// no key pressed

/* bool  Send_USB_packet(end point, pointer to buffer, length, toggle )   */
   Send_USB_packet (2,kb_tx_key,sizeof(kb_tx_key),TOGGLE_USB);

   //receive NUM LOCK, CAPS LOCK, etc LED status from PC.
  //we won't do anything with it.
if (Get_USB_Buffer(2))  // returns true if USB buffer is full
{
/* char Receive_USB_packet(endpoint, ptr to variable, #bytes to receive)
Receive_USB_packet(2, &LEDs, 1);
if (LEDs) AssertLEDs(&LEDs);// light LEDs specified
}
}
// In the USB Descriptor .h file:

///////// config options /////
#define  USB_CONFIG_PID       0x0280// Cirque PID for Easy Cat (Pinn AG)
#define  USB_CONFIG_VID       0x0488// Cirque VID
#define  USB_CONFIG_BUS_POWER 100   //100mA  (range is 0..500)
#define  USB_CONFIG_VERSION   0x0100    //01.00  //range is 00.00 to 99.99
/////// end config ///////////////////////////////////////////////////////////

const char USB_CLASS_SPECIFIC_DESC[] =
   {
      //hid report descriptor for interface 0 (mouse section)
      0x05, 0x01, // usage page (generic desktop Choose the usage page "mouse" is on   //0,1
      0x09, 0x02, // usage (mouse) Device is a mouse  //2,3
      0xA1, 0x01, // collection (application) This collection encompasses the report format  //4,5
      0x09, 0x01, // usage (pointer) Choose the key code usage page  //6,7
      0xA1, 0x00, // collection (physical) Physical collection //8,9
      0x05, 0x09, // usage page (buttons) Choose the “button” usage page   //10,11
      0x19, 0x01, // usage minimum (1) There are three buttons //12,13
      0x29, 0x03, // usage maximum (3) //14,15
      0x15, 0x00, // logical minimum (0) Each button is represented by one bit   //16,17
      0x25, 0x01, // logical maximum (1)  //18,19
      0x95, 0x03, // report count (3) Three reports, one bit each //20,21
      0x75, 0x01, // report size (1)   //22, 23
      0x81, 0x02, // input (data, variable, absolute) Defined bits above are data bits //24,25
      0x95, 0x01, // report count (1) One report, five bits in length   //26, 27
      0x75, 0x05, // report size (5)   //28, 29
      0x81, 0x01, // input (constant) Bit stuff to fill byte   //30, 31
      0x05, 0x01, // usage page (gen desktop) Choose the usage page “X” and “Y” are on   //32, 33
      0x09, 0x30, // usage (X) X direction of pointer //34, 35
      0x09, 0x31, // usage (Y) Y direction of pointer //36, 37
      0x09, 0x38  // usage (wheel)  //38, 39
      0x15, 0x81, // logical minimum (-127) Range of report data is -127 to 127  //40,41
      0x25, 0x7F, // logical maximum (127)   //42, 43
      0x75, 0x08, // report size (8) Two reports, eight bits each //44, 45
      0x95, 0x03, // report count (3)  //46, 47
      0x81, 0x06, // input (data, variable, absolute) Defined bits above are data bits //48, 49
      0xC0,       // end collection End physical collection //50
      0xC0        // end collection End application collection //51
      //hid report descriptor for interface 1 (keyboard section)
      0x05, 0x01, //usage page (generic desktop)   //52, 53
      0x09, 0x06, //usage (keyboard)   //54, 55
      0xA1, 0x01, //collection (application) //56, 57
      0x05, 0x07, //usage page (key codes)   //58, 59
      0x19, 0xE0, //usage min (224) //60, 61
      0x29, 0xE7, //usage max (231) //62, 63
      0x15, 0x00, //logical min (0) //64, 65
      0x25, 0x01, //logical max (1) //66, 67
      0x75, 0x01, //report size (1) //68, 69
      0x95, 0x08, //report count (8)   //70, 71
      0x81, 0x02, //input (data, variable, absolute) [modifier byte] //72, 73
      0x95, 0x01, //report count (1)   //74, 75
      0x75, 0x08, //report size (8)    //76, 77
      0x81, 0x01, //input (constant) [reserved byte]  //78, 79
      0x95, 0x05, //report count (5)   //80, 81
      0x75, 0x01, //report size (1)    //82, 83
      0x05, 0x08, //usage page (page# for leds) //84, 85
      0x19, 0x01, //usage min (1)   //86, 87
      0x29, 0x05, //usage max (5)   //88, 89
      0x91, 0x02, //output (data, var, abs) [led report] //90, 91
      0x95, 0x01, //report count (1)   //92, 93
      0x75, 0x03, //report size (3) //94, 95
      0x91, 0x01, //output (constant) [led report padding]  //96, 97
      0x95, 0x05, //report count (5)   //98, 99
      0x75, 0x08, //report size (8) //100, 101
      0x15, 0x00, //logical min (0) //102, 103
      0x25, 0x65, //logical max (101)  //104, 105
      0x05, 0x07, //usage page (key codes)   //106, 107
      0x19, 0x00, //usage min (0)   //108, 109
      0x29, 0x65, //usage max (101) //110, 111
      0x81, 0x00, //input (data, array)   //112, 113
      0xC0        //end collection  //114
   };

const char USB_CONFIG_DESC[] =
{
   // The Order of this Array must be the following:
      //    config(s)
      //    interface(s)
      //    class(es)
      //    endpoint(s)

   //config_descriptor for config index 1
        USB_DESC_CONFIG_LEN, //length of descriptor size          ==0
        USB_DESC_CONFIG_TYPE, //constant CONFIGURATION (CONFIGURATION 0x02)     ==1
        USB_TOTAL_CONFIG_LEN,0, //size of all data returned for this config      ==2,3
        2, //number of interfaces this device supports       ==4
       0x01, //identifier for this configuration.  (IF we had more than one configurations)     ==5
        0x00, //index of string descriptor for this configuration      ==6
        0x80, //bit 6=1 if self powered    ==7
        0x32, //max bus power required (0x32 = 100mA)

   //interface descriptor 1 (MOUSE)
         USB_DESC_INTERFACE_LEN, //length of descriptor      =9
         USB_DESC_INTERFACE_TYPE, //constant INTERFACE (INTERFACE 0x04)       =10
         0x00, //number defining this interface (IF we had more than one interface)    ==11
         0x00, //alternate setting     ==12
         1, //number of endpoints for this interface  ==13
         0x03, //class code, 03 = HID     ==14
         0x01, //subclass code //boot     ==15
         0x02, //protocol code (mouse)      ==16
         0x00, //index of string descriptor for interface      ==17

   //class descriptor 1  (HID)
         USB_DESC_CLASS_LEN, //length of descriptor    ==18
         USB_DESC_CLASS_TYPE, //dscriptor type (0x21 == HID)  ==19
         0x00,0x01, //hid class release number (1.0)      ==20,21
         0x00, //localized country code (0 = none)       ==22
         0x01, //number of hid class descrptors that follow (1)     ==23
         0x22, //report descriptor type (0x22 == HID)                ==24
         USB_CLASS_SPECIFIC_DESC_LOOKUP_SIZE[0][0], 0x00, //length of report desc ==25,26

   //endpoint descriptor 1
         USB_DESC_ENDPOINT_LEN, //length of descriptor                   ==27
         USB_DESC_ENDPOINT_TYPE, //constant ENDPOINT (ENDPOINT 0x05)          ==28
         0x81, //endpoint number and direction (0x81 = EP1 IN)       ==29
         USB_ENDPOINT_TYPE_INTERRUPT, //transfer type supported (0x03 is interrupt)         ==30
         USB_EP1_TX_SIZE,0x00, //maximum packet size supported                  ==31,32
         10,  //polling interval, in ms.  (cant be smaller than 10 for slow speed devices)     ==33

   //interface descriptor 2 (KEYBOARD)
         USB_DESC_INTERFACE_LEN, //length of descriptor      =34
         USB_DESC_INTERFACE_TYPE, //constant INTERFACE (INTERFACE 0x04)       =35
         0x01, //number defining this interface (IF we had more than one interface)    ==36
         0x00, //alternate setting     ==37
         2, //number of endpoints for this interface  //38
         0x03, //class code, 03 = HID     ==39
         0x01, //subclass code //boot     ==40
         0x01, //protocol code (keyboard)      ==41
         0x00, //index of string descriptor for interface      ==42

   //class descriptor 2  (HID)
         USB_DESC_CLASS_LEN, //length of descriptor    ==43
         USB_DESC_CLASS_TYPE, //dscriptor type (0x21 == HID)      ==44
         0x00,0x01, //hid class release number (1.0) (try 1.10)      ==45,46
         0x00, //localized country code (0 = none)       ==47
         0x01, //number of hid class descrptors that follow (1)     ==48
         0x22, //report descriptor type (0x22 == HID)                ==49
         USB_CLASS_SPECIFIC_DESC_LOOKUP_SIZE[0][1], 0x00, //length of report desc ==50,51

   //endpoint descriptor 2 IN
         USB_DESC_ENDPOINT_LEN, //length of descriptor                   ==52
         USB_DESC_ENDPOINT_TYPE, //constant ENDPOINT (ENDPOINT 0x05)          ==53
         0x82, //endpoint number and direction (0x82 = EP2 IN)       ==54
         USB_ENDPOINT_TYPE_INTERRUPT, //transfer type supported (0x03 is interrupt)         ==55
         USB_EP2_TX_SIZE,0x00, //maximum packet size supported                  ==56,57
         10,  //polling interval, in ms.  (cant be smaller than 10 for slow speed devices)     ==58

   //endpoint descriptor 2 OUT
         USB_DESC_ENDPOINT_LEN, //length of descriptor                   ==59
         USB_DESC_ENDPOINT_TYPE, //constant ENDPOINT (ENDPOINT 0x05)          ==60
         0x02, //endpoint number and direction (0x81 = EP1 IN)       ==61
         USB_ENDPOINT_TYPE_INTERRUPT, //transfer type supported (0x03 is interrupt)         ==62
         USB_EP2_RX_SIZE,0x00, //maximum packet size supported                  ==63,64
         10  //polling interval, in ms.  (cant be smaller than 10 for slow speed devices)     ==65
};
```

11. Connecting the Cirque Driver to a Composite Device

This section describes how to connect the Cirque driver, which includes:

- Installing the driver.
- Starting the Device Manager.
- Scanning With the Device Manager.
- Finding the VID and PID.

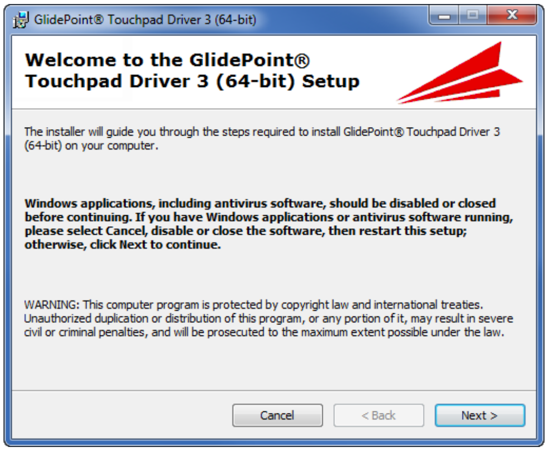
12. Installing the Cirque Driver

This section provides the installation steps for a PC with Windows 7 installed.

Follow these steps to install the Cirque driver.

1. Insert the Cirque GlidePoint® software CD into your CD-ROM.

   If the AutoRun feature does not launch the Windows Installer, select RUN from the START menu and type D:\splash.exe (where D is the drive letter of your CD-ROM). The Welcome screen will then appear.



GlidePoint® Touchpad Driver 3 (64-bit)

Welcome to the GlidePoint® Touchpad Driver 3 (64-bit) Setup

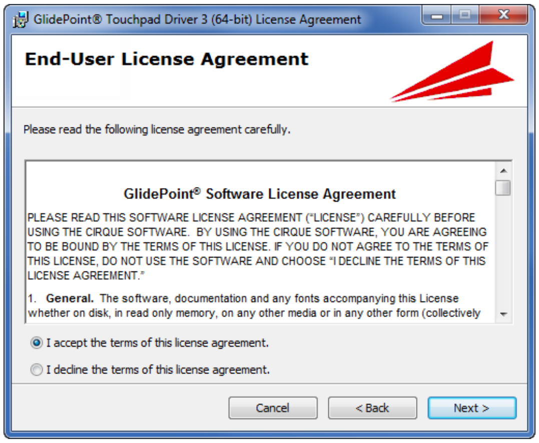
The installer will guide you through the steps required to install GlidePoint® Touchpad Driver 3 (64-bit) on your computer.

Windows applications, including antivirus software, should be disabled or closed before continuing. If you have Windows applications or antivirus software running, please select Cancel, disable or close the software, then restart this setup; otherwise, click Next to continue.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Figure 2. GlidePoint Driver Installation Welcome Screen

2. Click the **Next>** button. The End-User License Agreement screen will then appear.



GlidePoint® Touchpad Driver 3 (64-bit) License Agreement

End-User License Agreement

Please read the following license agreement carefully.

GlidePoint® Software License Agreement

PLEASE READ THIS SOFTWARE LICENSE AGREEMENT ("LICENSE") CAREFULLY BEFORE USING THE CIRQUE SOFTWARE. BY USING THE CIRQUE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, DO NOT USE THE SOFTWARE AND CHOOSE "I DECLINE THE TERMS OF THIS LICENSE AGREEMENT."

1. **General.** The software, documentation and any fonts accompanying this License whether on disk, in read only memory, on any other media or in any other form (collectively

● I accept the terms of this license agreement.

○ I decline the terms of this license agreement.

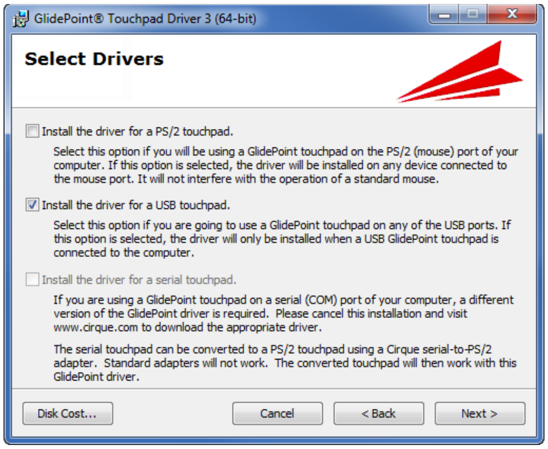
Figure 3. End-User License Agreement Screen

3. Read the License agreement.

4. Select the **I accept the terms of the license agreement** radio button and then click the **Next>** button.

Note: *If you do not accept the terms of the agreement, contact your Cirque representative.*

The Select Drivers screen will then appear.



GlidePoint® Touchpad Driver 3 (64-bit)

Select Drivers

□ Install the driver for a PS/2 touchpad.

Select this option if you will be using a GlidePoint touchpad on the PS/2 (mouse) port of your computer. If this option is selected, the driver will be installed on any device connected to the mouse port. It will not interfere with the operation of a standard mouse.

☑ Install the driver for a USB touchpad.

Select this option if you are going to use a GlidePoint touchpad on any of the USB ports. If this option is selected, the driver will only be installed when a USB GlidePoint touchpad is connected to the computer.

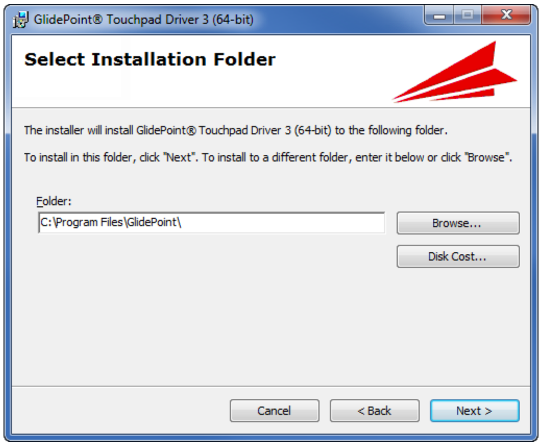
□ Install the driver for a serial touchpad.

If you are using a GlidePoint touchpad on a serial (COM) port of your computer, a different version of the GlidePoint driver is required. Please cancel this installation and visit www.cirque.com to download the appropriate driver.

The serial touchpad can be converted to a PS/2 touchpad using a Cirque serial-to-PS/2 adapter. Standard adapters will not work. The converted touchpad will then work with this GlidePoint driver.

Figure 4. The Select Drivers Screen

5. Select the **Install the driver for a USB touchpad** check box and then click the **Next>** button. The Select Installation Folder screen will then appear.



GlidePoint® Touchpad Driver 3 (64-bit)

Select Installation Folder

The installer will install GlidePoint® Touchpad Driver 3 (64-bit) to the following folder.

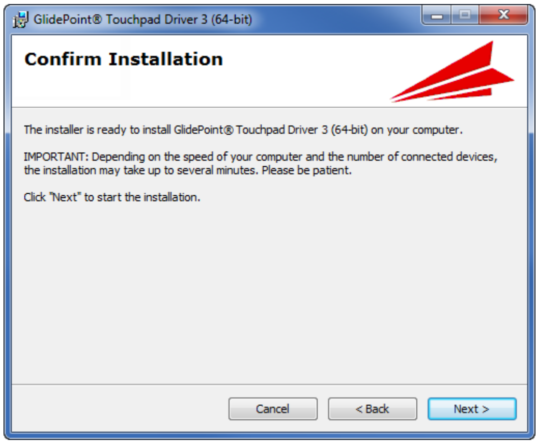
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Program Files\GlidePoint\

Figure 5. The Select Installation Folder Screen

Unless you specify a different directory, the driver will be installed in the C:\Program Files\GlidePoint\ directory (where C: is your computer hard drive.)

6. Click the **Next >** button. The Confirm Installation screen will then appear.



GlidePoint® Touchpad Driver 3 (64-bit)

Confirm Installation

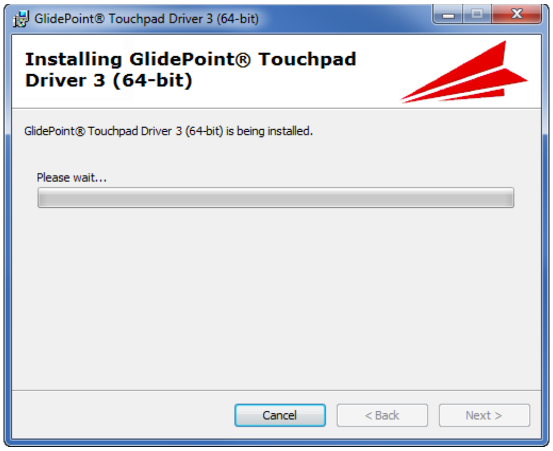
The installer is ready to install GlidePoint® Touchpad Driver 3 (64-bit) on your computer.

IMPORTANT: Depending on the speed of your computer and the number of connected devices, the installation may take up to several minutes. Please be patient.

Click "Next" to start the installation.

Figure 6. The Confirm Installation Screen

7. Click the **Next >** button. The Installation process will then begin. A progress bar will indicate the current installation status.



GlidePoint® Touchpad Driver 3 (64-bit)

Installing GlidePoint® Touchpad Driver 3 (64-bit)

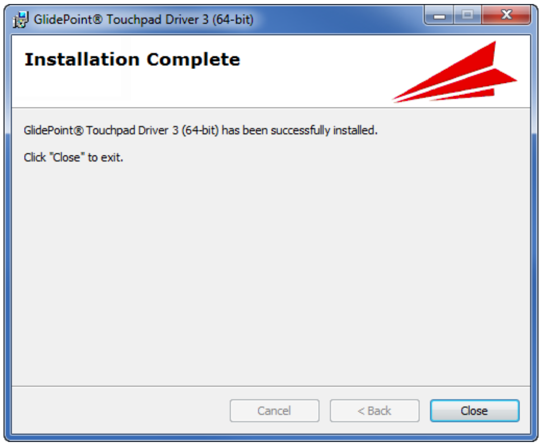
GlidePoint® Touchpad Driver 3 (64-bit) is being installed.

Please wait...

Figure 7. The Confirm Installation Screen

Note: *Allow any alerts that may appear.*

The installation Complete screen will then appear.



GlidePoint® Touchpad Driver 3 (64-bit)

Installation Complete

GlidePoint® Touchpad Driver 3 (64-bit) has been successfully installed.

Click "Close" to exit.

Figure 8. The Installation Complete Screen

Click the **Close** button. Re-boot your computer if prompted to do so.

13. Starting the Device Manager

The Device Manager identifies the devices installed on your computer. It can also be used to update driver software, to check to see if hardware is working properly, and to modify hardware settings.

The Device Manager interface is accessed through the System section of the Control panel. There may be different steps depending on your computer settings.

The Device Manager MUST be started before connecting the USB composite device to the PC.

13.1 Starting the Device Manager Interface

Follow these steps to open the Device Manager interface.

1. Navigate to the System Properties:

   Start menu\Control Panel\All Control Panel Items\System

   Or

   Press the **Windows** + **Pause/break** key combination on your keyboard.

2. Click the **Device Manager** hyperlink in the left panel (Figure 8).

Click this link

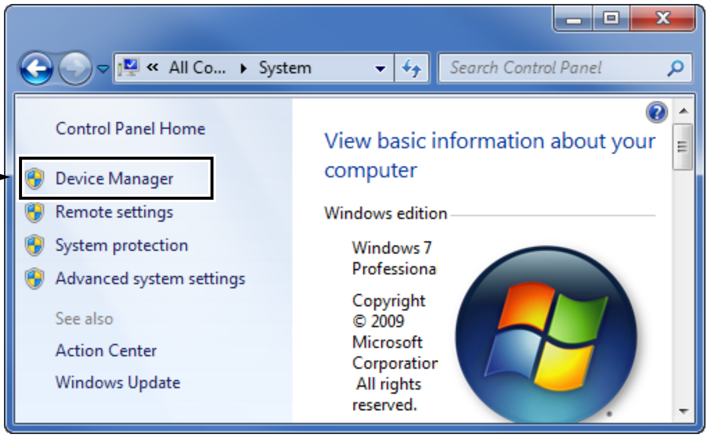


Figure 9. The Device Manager Hyperlink

The Device Manager will then appear (Figure 10 on page 29).

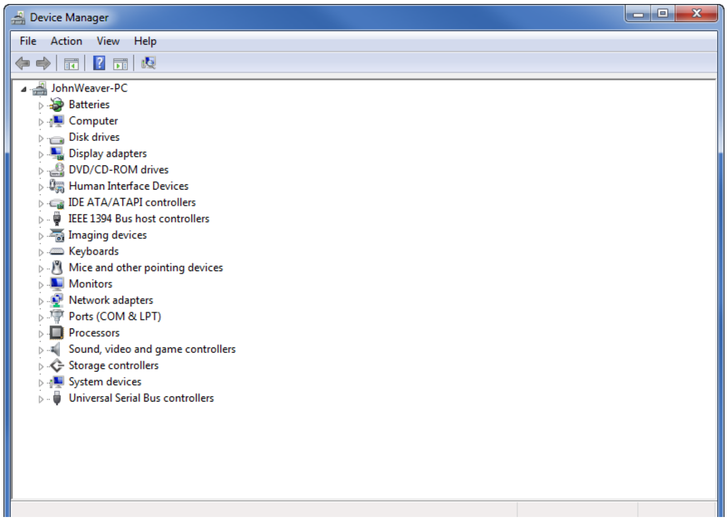


Figure 10. The Device Manager

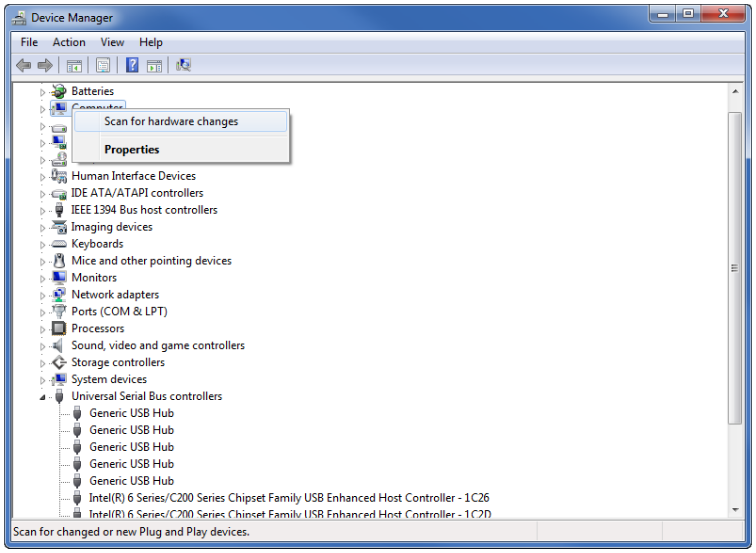
Connect the USB composite device to the USB port on the PC.

14. Scanning With the Device Manager

The device manager must find and recognize the device.

Follow these steps to scan for the device within the Device Manager:

1. Right-click the Computer icon and then select the Scan for hardware changes option (Figure 11).


Device Manager
File
Action
View
Help
Batteries
Computer
Scan for hardware changes
Properties
Human Interface Devices
IDE ATA/ATAPI controllers
IEEE 1394 Bus host controllers
Imaging devices
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Processors
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers
Generic USB Hub
Generic USB Hub
Generic USB Hub
Generic USB Hub
Generic USB Hub
Intel(R) 6 Series/C200 Series Chipset Family USB Enhanced Host Controller - 1C26
Intel(R) 6 Series/C200 Series Chipset Family USB Enhanced Host Controller - 1C2D

Scan for changed or new Plug and Play devices.

Figure 11. Selecting Scan for hardware changes

2. Select the **Devices by connection** option from the View menu (View> Devices by connection).

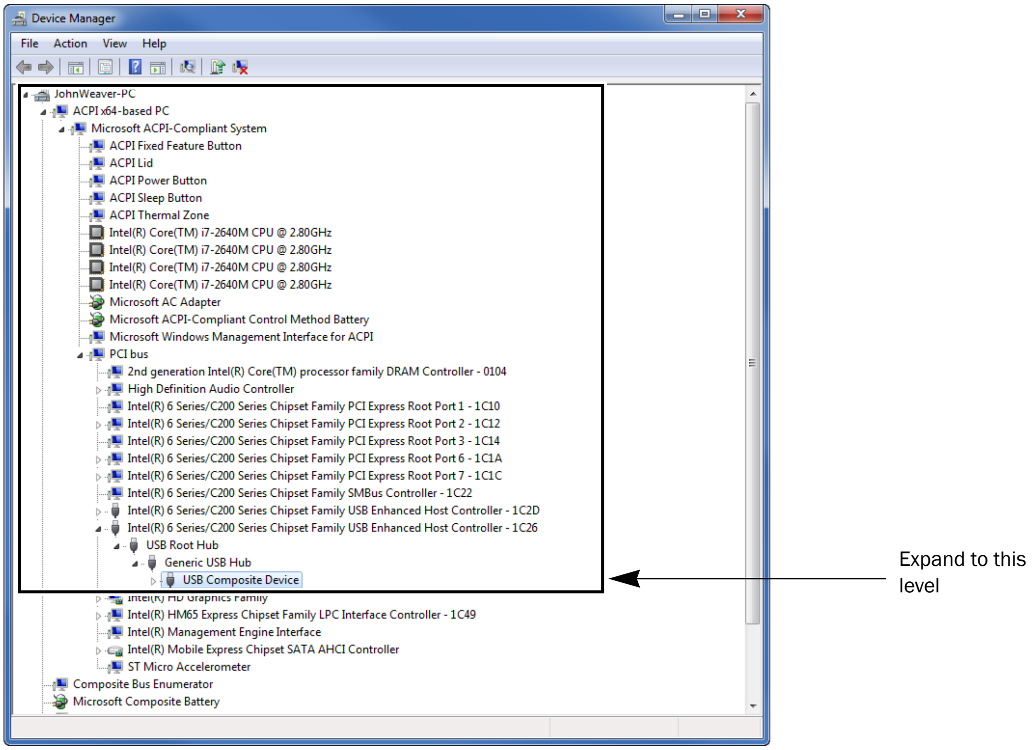
3. Navigate to the USB Composite device by:

   a. Click the arrow next to the **ACPI x64-based PC' (or ACPI ... PC)** option to expand the selection.

   b. Expand the **Microsoft ACPI-Compliant System** option.

   c. Expand the **PCI bus** option.

   Look for the USB connector ICON or the letters USB in the description. There may be more than one descriptor to search. Each descriptor must be searched until your company's VID and PID is found (Figure 12 on page 31).



```
JohnWeaver-PC
  ACPI x64-based PC
    Microsoft ACPI-Compliant System
      ACPI Fixed Feature Button
      ACPI Lid
      ACPI Power Button
      ACPI Sleep Button
      ACPI Thermal Zone
      Intel(R) Core(TM) i7-2640M CPU @ 2.80GHz
      Intel(R) Core(TM) i7-2640M CPU @ 2.80GHz
      Intel(R) Core(TM) i7-2640M CPU @ 2.80GHz
      Intel(R) Core(TM) i7-2640M CPU @ 2.80GHz
      Microsoft AC Adapter
      Microsoft ACPI-Compliant Control Method Battery
      Microsoft Windows Management Interface for ACPI
      PCI bus
        2nd generation Intel(R) Core(TM) processor family DRAM Controller - 0104
        High Definition Audio Controller
        Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 1 - 1C10
        Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 2 - 1C12
        Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 3 - 1C14
        Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 6 - 1C1A
        Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 7 - 1C1C
        Intel(R) 6 Series/C200 Series Chipset Family SMBus Controller - 1C22
        Intel(R) 6 Series/C200 Series Chipset Family USB Enhanced Host Controller - 1C2D
        Intel(R) 6 Series/C200 Series Chipset Family USB Enhanced Host Controller - 1C26
          USB Root Hub
            Generic USB Hub
              USB Composite Device
        Intel(R) HD Graphics Family
        Intel(R) HM65 Express Chipset Family LPC Interface Controller - 1C49
        Intel(R) Management Engine Interface
        Intel(R) Mobile Express Chipset SATA AHCI Controller
        ST Micro Accelerometer
  Composite Bus Enumerator
  Microsoft Composite Battery
```

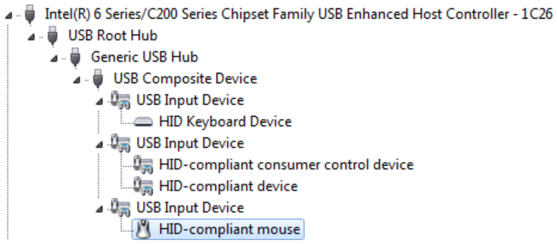
Expand to this level

Figure 12. Selecting Scan for hardware changes

For Example: Intel® 6 Series/C200 Series Chipset Family USB Enhanced Host Controller - 1C26.

d. Expand this descriptor. Expand any Generic Root Hub or USB Hub s, until you locate a USB Composite Device.

e. Expand each USB Input Device descriptor to discover your HID-Compliant devices.



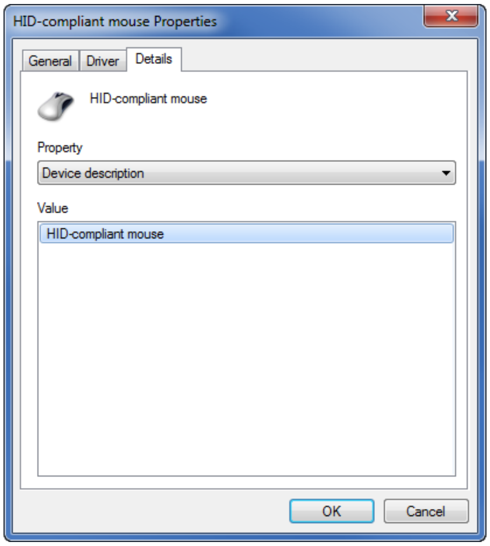
```
Intel(R) 6 Series/C200 Series Chipset Family USB Enhanced Host Controller - 1C26
  USB Root Hub
    Generic USB Hub
      USB Composite Device
        USB Input Device
          HID Keyboard Device
        USB Input Device
          HID-compliant consumer control device
          HID-compliant device
        USB Input Device
          HID-compliant mouse
```

Figure 13. HID-Compliant devices

4. Double click on the **USB Input Device** that describes the HID-compliant mouse. The HID-compliant mouse Properties dialog will then appear.

15. Setting the VID and PID For the Device

1. Select the Details Tab (Figure 14).



Property

Device description

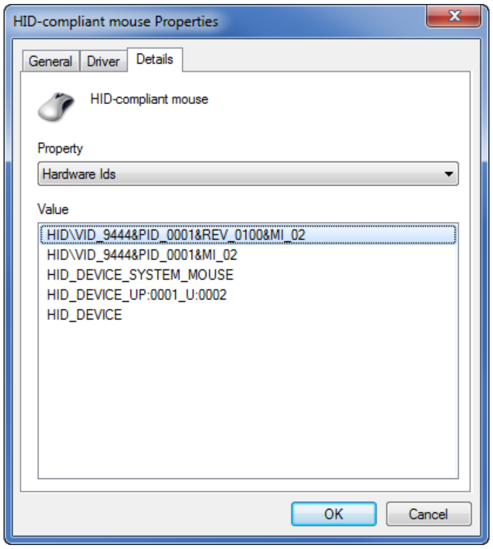
Value

HID-compliant mouse

Figure 14. he HID-compliant Mouse Properties Dialog

2. Click the **Properties** drop-down menu in the middle of the dialog.

3. Select the **Hardware Ids** option from the list. This information will appear in the Value section (Figure 15).



Property

Hardware Ids

Value

```
HID\VID_9444&PID_0001&REV_0100&MI_02
HID\VID_9444&PID_0001&MI_02
HID_DEVICE_SYSTEM_MOUSE
HID_DEVICE_UP:0001_U:0002
HID_DEVICE
```

Figure 15. he HID-compliant Mouse Properties Dialog - with Hardware Id Values

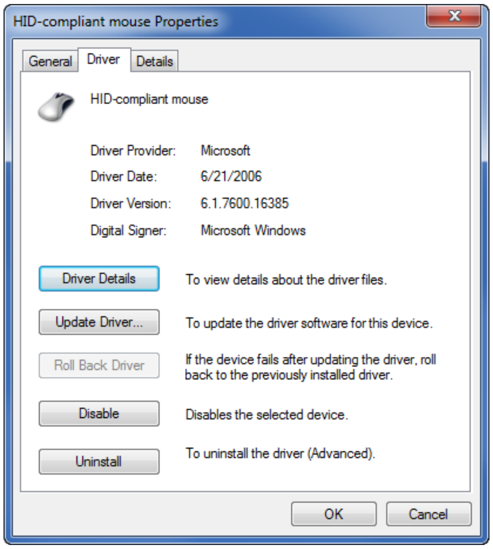
4. Check the USB\VID for your number the full text especially the &MI_xx number. For example: USB\VID_9999D&PID_5555&REV_100&MI_01

Note: *If this device is not yours, repeat the steps described in Scanning With the Device Manager on page 30, until you locate your HID-compliant mouse VID, PID, and child number.*

15.1 Updating the Driver

Follow these steps to update the driver:

1. Select the Descriptor in the Value field.
2. Click the **Driver** Tab (Figure 16).



HID-compliant mouse

Driver Provider: Microsoft
Driver Date: 6/21/2006
Driver Version: 6.1.7600.16385
Digital Signer: Microsoft Windows

To view details about the driver files.

To update the driver software for this device.

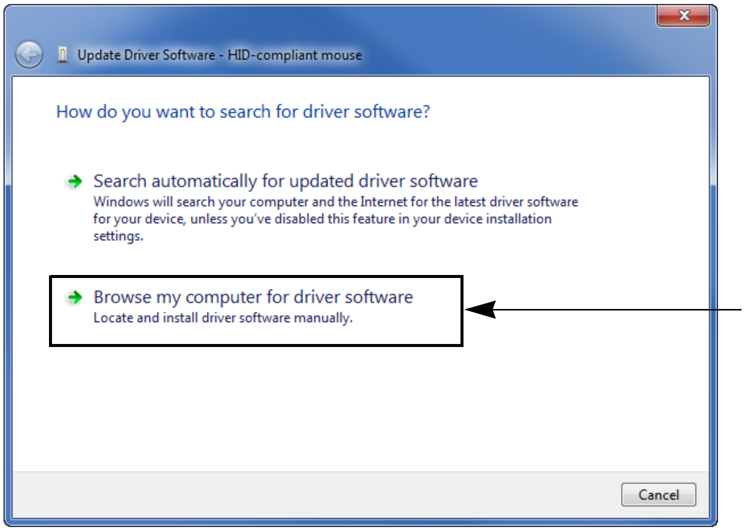
If the device fails after updating the driver, roll back to the previously installed driver.

Disables the selected device.

To uninstall the driver (Advanced).

Figure 16. The HID-compliant Mouse Properties Dialog - Driver Tab

3. Select the **Update Driver** button. The Update driver interface will then appear.



How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

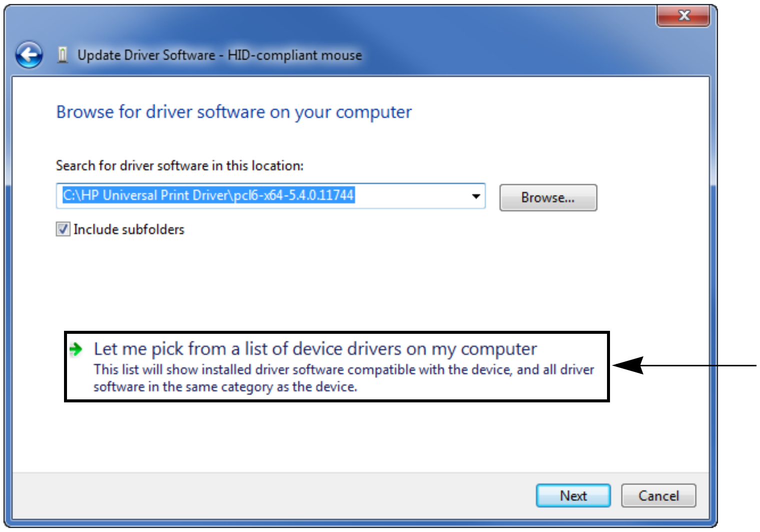
Browse my computer for driver software
Locate and install driver software manually.

Click this option

Figure 17. The Update Driver Interface

4. Click the **Browse my computer for driver software** option (Figure 17 on page 33).

   The Browse for driver software on my computer screen will then appear.



Browse for driver software on your computer

Search for driver software in this location:

C:\HP Universal Print Driver\pcl6-x64-5.4.0.11744

☑ Include subfolders

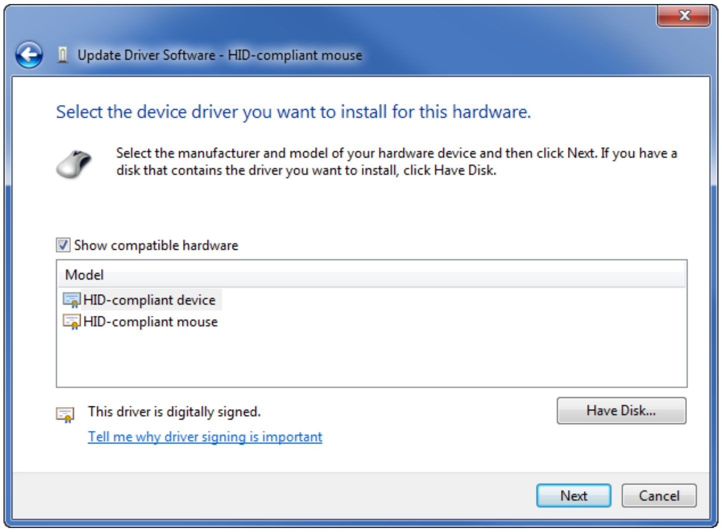
Let me pick from a list of device drivers on my computer

This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Click this option

Figure 18. The Browse for Driver Software On My Computer Screen

Click the **Let me pick from a list of device driver on my computer** option. The Select the device driver you want to install for this hardware screen will then appear.



Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

☑ Show compatible hardware

Model

HID-compliant device

HID-compliant mouse

This driver is digitally signed.

Tell me why driver signing is important

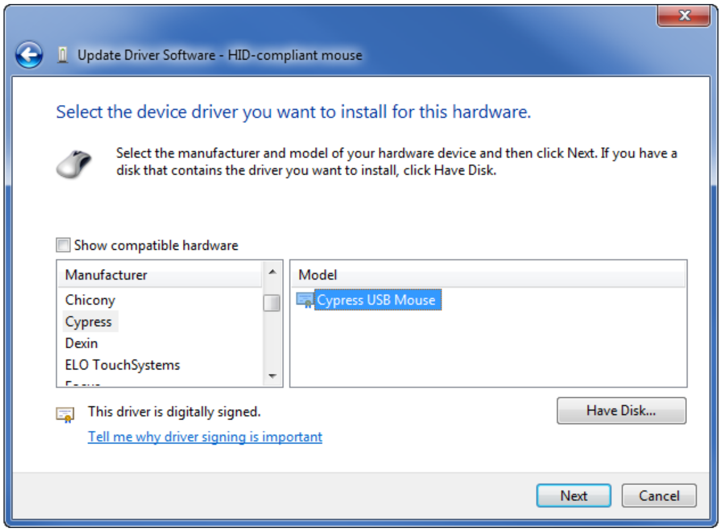
Figure 19. The Browse for Driver Software On My Computer Screen

IMPORTANT: Deselect the **Show compatible hardware** *check box above the Model selection box.*

The interface will then change to show two columns: Manufacture and Model.

5. Scroll down the *Manufacturer* selection box to find and select **Cypress.**

6. Select **Cypress USB Mouse** in the *Model* selection box (Figure 20 on page 35).



Update Driver Software - HID-compliant mouse

Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

□ Show compatible hardware

Manufacturer	Model
Chicony	Cypress USB Mouse
Cypress	
Dexin	
ELO TouchSystems	

This driver is digitally signed.

Tell me why driver signing is important

Figure 20. Selecting the Cypress USB Mouse Option

7. Click the **Next** button. Allow any warning you receive.

Windows should remember this driver next time you plug in your USB composite device.

16. Contact Information

Contact a Cirque sales representative for a complete list of Cirque's OEM products.

In United States & Canada	(800) GLIDE-75 (454-3375)
Outside US & Canada	(801) 467-1100
Fax	(801) 467-0208
Web site	http://www.cirque.com

16.0.1 Part Ordering Information

When ordering this part, please contact your Cirque representative to assist you in selecting the correct size, power, configurations, and overlay that will best meet your trackpad needs.

THIS INFORMATION IS PROVIDED "AS IS." CIRQUE SPECIFICALLY DISCLAIMS ALL WARRANTIES EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENTATION AND USE OF THE DOCUMENTATION IN DESIGN OF ANY PRODUCTS OR FOR ANY PARTICULAR APPLICATION.

16.0.1.1References:

"www.usb.org : USB Descriptors and examples

"www.usb.org HID_Usage_Tables show the mouse packet format keyboard key values

"www.microchip.com or www.cypress.com : USB information, training and code examples

"See the *Sample Code* on page 20 for code examples