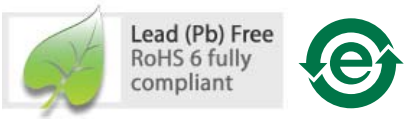
ADNK-5053-HT01

Optical Mouse Designer's Kit

Design Guide



Introduction

The Universal Serial Bus (USB) is an industry standard serial interface between a computer and peripherals such as a mouse, joystick, keyboard, UPS, etc. This design guide describes how a cost-effective USB optical mouse can be built using Avago Technologies mainstream, small form factor ADNS-5050 optical mouse sensor and Holtek Semiconductor HT82M99E USB microcontroller. The document starts with the basic operations of a computer mouse peripheral followed by an introduction to the Avago Technologies ADNS-5050 Optical Navigation Sensor and HT82M99E USB microcontroller. A schematic of the HT-82M99E USB microcontroller to the ADNS-5050 optical mouse sensor and buttons of a standard mouse can be found in Appendix A. The software section of this application note describes the architecture of the firmware required to implement the USB mouse functions. The ADNS-5050 data sheet is available from the Avago web site at http://www.avagotech.com/. The HT82M99E data sheet is available from the Holtek web site at www.holtek.com. USB documentation can be found at the USB Implementers Forum web site at www.usb.org.

Optical Mouse Basics

The optical mouse measures changes in position by optically acquiring sequential surface images (frames), and mathematically determining the direction and magnitude of movement. The traditional dual-channel mechanical encoder generates the quadrature Z-wheel movement signals. This design guide shows how to connect to and manage a standard configuration of mouse hardware, as well as handle the USB protocols. Each of these protocols provides a standard way of reporting mouse movement and button presses to the PC.

Introduction to ADNS-5050 Optical Mouse Sensor

Avago's ADNS-5050 optical sensor is used in this reference design as the primary navigation engine. This Optical Navigation Technology contains an Image Acquisition System, a Digital Signal Processor, and a three-wire serial port. The HT82M99E periodically reads the ADNS-5050's Delta_X and Delta_Y registers to obtain any horizontal and vertical motion information happening as a result of the mouse being moved. The three-wire synchronous serial port is used to set and read parameters in the ADNS-5050, and to read out the motion, (delta) X and (delta) Y information.

This motion information will be reported to the PC to update the position of the cursor. The advantages of using ADNS-5050 optical sensor are: good tracking accuracy, small form factor, sensor programming flexibility via SPI port, and the automatic frame rate feature. There is also a wide range of selectable sensor resolution between 125cpi to 1375cpi. Furthermore, ADNS-5050 sensor has built-in oscillator and on-chip LED driver to minimize external components. Additionally, Burst mode is another special serial port operation mode which may be used to reduce the serial transaction time for motion read operation. By reading the Motion_Burst register, ADNS-5050 will respond with the contents of the Delta_X, Delta_Y, SQUAL, Shutter_Upper, Shutter_Lower, Maximum_Pixel and Pixel_Sum registers in that order.

To learn more about sensor's technical information, please visit the Avago web site at http://www.avagotech.com/

Mechanical Z-Wheel

The motion of Z-wheel is detected using the traditional method by decoding the quadrature signal generated by mechanical encoder. The Z-pinwheel is connected to the Z-encoder through its shaft. The rotational movement of the shaft is decoded into on and off levels in a quadrature output pattern. Every change in the Z-encoder outputs represents a count of mouse movement. Comparing the last state of the Z-encoder to the current state derives direction information. As shown in Figure 1, traveling in clockwise direction produces a unique set of state transitions, and traveling in counter clockwise direction produces another set of unique state transitions. In this reference design, only the motion at the Z-wheel is detected using this method.

Mouse Buttons

Mouse buttons are connected as standard switches. These switches are pulled up by the pull up resistors inside the microcontroller. When the user presses a button, the switch will be closed and the pin will be pulled LOW to GND. A LOW state at the pin is interpreted as the button being pressed. A HIGH state is interpreted as the button has been released or the button is not being pressed. Normally the switches are debounced in firmware for 15-20ms. In this reference design there are five switches: left, middle, right, CPI+ and CPI-.

Introduction to the Holtek HT82M99E

The HT82M99E is an 8-bit RISC microcontroller with an integrated USB Serial Interface Engine (SIE). The architecture executes general-purpose instructions that are optimized for USB applications. The HT82M99E has a built-in timers as well as programmable drive strength and pull-up resistors on each I/O line. High performance, low-cost human interface type computer peripherals can be implemented with a minimum of external components and firmware effort.

Serial Peripheral Interface (SPI)

The HT82M99E does not have integrated SPI circuit. The SPI function is accomplished by utilizing GPIO pin and firmware.

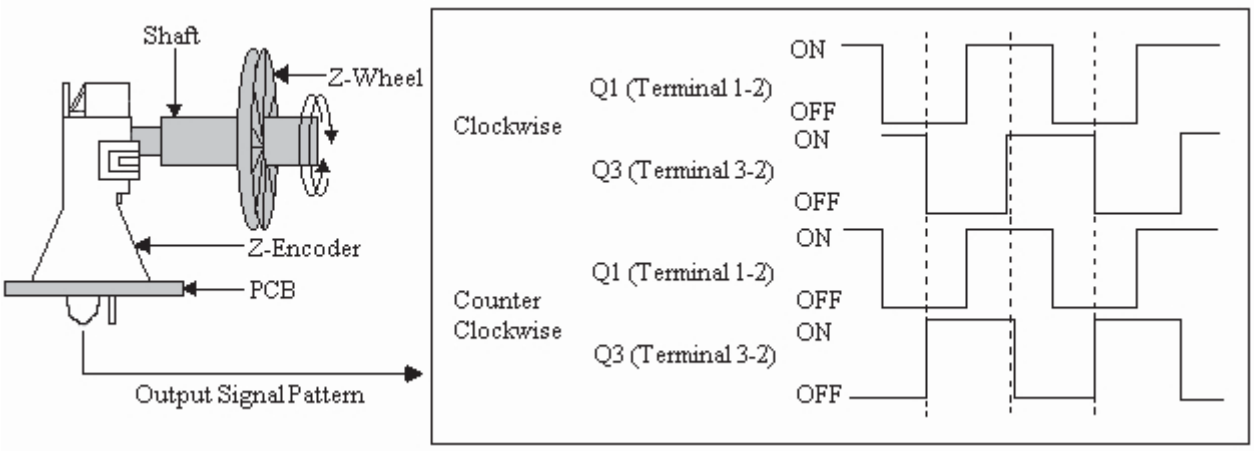


Figure 1. Mechanical Z-Wheel Output Signal Generation

Hardware Implementation

The standard hardware to implement a mouse is shown in Figure 2. For X and Y movement, the optical sensor is used. The Z- wheel movement is detected by a set of mechanical encoders that output quadrature signals. For each button there is a switch that is pulled up internally by the built in pull up resistors.

Firmware Configurable GPIO

The reference firmware is configured to use the GPIO pins as shown on the schematic in Appendix A. However, it may be more optimal to use a different I/O configuration to meet the mechanical constraints of PCB design. The reference firmware is designed to be easily configured to another set of pin connections. This is accomplished through changes in the I/O definitions in the firmware which can be found in the ADNK-5053-HT01 CD. The following statements are the pin definitions as they exist today. The firmware will use these definitions to read and configure the GPIO pins, without any other modifications.

Communications between the HT82M99E and the ADNS-5050 are done through MCU GPIO and firmware. The serial port cannot be activated while the chip is in reset (NRESET low). When the SPI is enabled thru PA7, PA0 and PA1 GPIO pins serve special functions enabling the SPI interface to talk with external hardware (sensor). During normal operation, the HT82M99E SPI acts as a Master to output the serial clock on PA0. Therefore, the USB microcontroller always initiates communication. Data to and from the ADNS-5050 optical sensor happens on PA1.

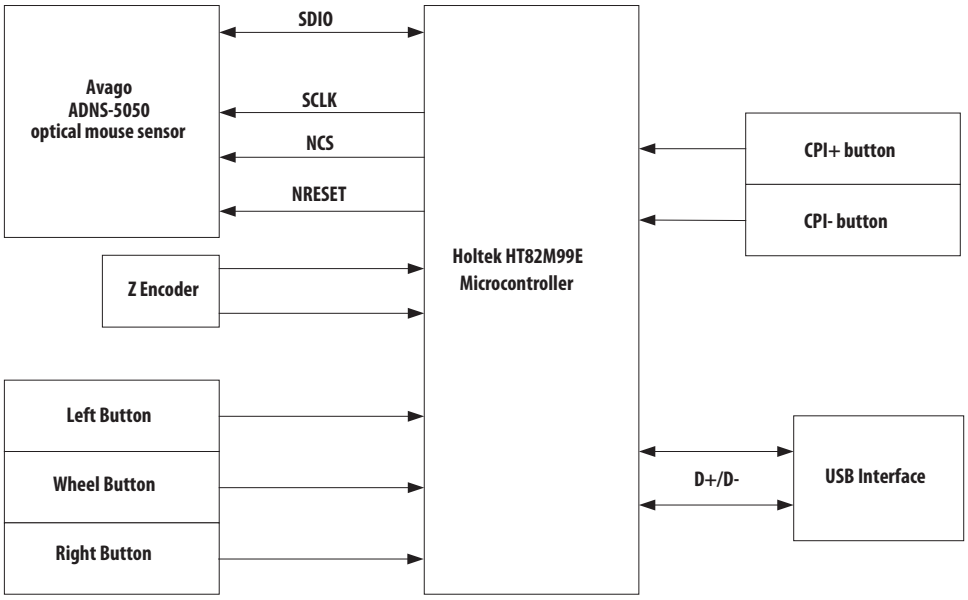


Figure 2. ADNS-5050 - HT82M99E Optical Mouse Hardware Block Diagram

ADNK-5053-HT01 Designer's Kit – Optical Mouse

The ADNK-5053-HT01 reference design mouse unit allows users to evaluate the performance of the Optical Tracking Engine (sensor, lens, LED assembly clip, LED) over a USB connection, using a Holtek USB Controller. This kit also enables users to understand the recommended mechanical assembly. (See Appendix C, D, and E)

System Requirements

PCs using Windows® 95/ Windows® 98/ Windows® NT/ Windows® 2000/ Windows® XP/ Windows® Vista with standard 3-button USB mouse driver loaded.

Functionality

3-button, scroll wheel optical mouse.

Operating

Hot pluggable with USB port. The PC does not need to be powered off when plugging or unplugging the evaluation mouse.

To Disassemble the ADNK-5053-HT01 Unit

The ADNK-5053-HT01 comprises of the plastic mouse casing, printed circuit board (PCB), lens, buttons, and USB cable. (See Figure 3.) Unscrewing the one screw located at the base of the unit can open the ADNK-5053-HT01 unit. Lifting and pulling the PCB out of the base plate can further disassemble the mouse unit.

Caution: The lens is not permanently attached to the sensor and will drop out of the assembly.

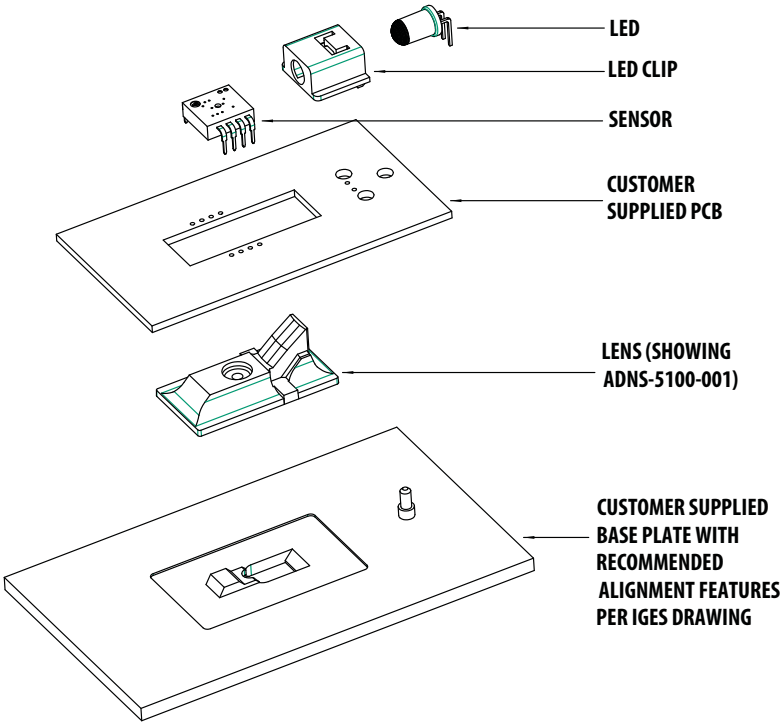


Figure 3. Exploded view drawing of optical tracking engine with ADNS-5050 optical mouse sensor.

While reassembling the components, please make sure that the Z height (Distance from lens reference plane to surface) is valid. Refer to Figure 4.

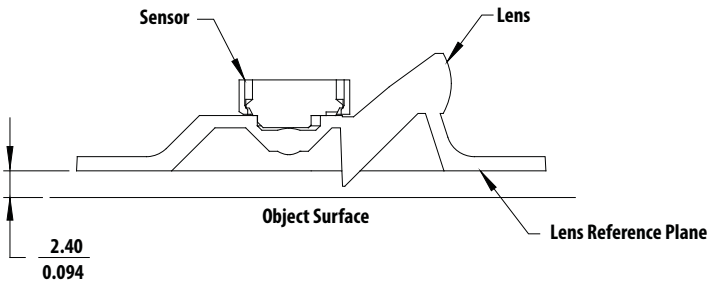


Figure 4. Distance from lens reference plane to surface

Below is the summary of the components contained in the ADNK-5053-HT01 Designer's Kit.

Sensor

The sensor technical information is contained in the ADNS-5050 Data Sheet.

USB Controller

Technical information on the Holtek USB controller is contained in the HT82M99E Data Sheet. The enclosed ADNK-5053-HT01 CD-ROM contains some firmware examples which designers can use to make changes and recompile. For further information on this product, please contact Holtek Semiconductor.

Lens

The lens technical information is contained in the ADNS-5100-001 Trim Lens Data Sheet. The flange on the standard ADNS-5100 Trim Lens is for ESD protection.

LED

The LED technical information is contained in the HLMP-EG3E-xxxxx Data Sheet.

Base Plate Feature – IGES File

The IGES file on the CD-ROM provides recommended base plate molding features to ensure optical alignment. This includes PCB assembly diagrams like solder fixture in assembly and exploded view, as well as solder plate. See Appendix D for details.

Reference Design Documentation – Gerber File

The Gerber File presents detailed schematics used in ADNK-5053-HT01 in PCB layout form. See Appendix C for more details.

Overall circuit

A schematic of the overall circuit is shown in Appendix A of this document. Appendix B lists the bill of materials.

Firmware Implementation

The firmware for this reference design is written in the assembly language. The following files are required to compile the mouse firmware:

HT82M99E.INC – the HT82M99E I/O registers definition.

ANTIEFT.LIB – the library code for EFT noise recovery

MAIN1.ASM – main mouse firmware

CLS1.ASM – USB HID class request routine

DES2.ASM – all mouse descriptor

HP_SENSOR.ASM – sensor assessing, button check, wheel check function

START.ASM – program start point

RF_VAR.ASM – user variables definition

STD1.ASM – USB standard request routine

USB_INT.ASM – USB interrupt routine

USBLIB1.ASM – USB FIFO access routine

PROTECTEDDATA.ASM – user RAM cannot be cleared by USB reset

At power up, the firmware examines the host interface and determines if the mouse is plugged into a USB host connection. After the interface type has been determined, the host firmware configures itself to operate on the detected interface. In ADNK-5053-HT01, only USB interface will be detected.

USB Interface

All USB Human Interface Device (HID) class applications follow the same USB start-up procedure. The procedure is as follows

1. **Device Plug-in**

   When a USB device is first connected to the bus, it is powered and running firmware, but communications on the USB remain non-functional until the host has issued a USB bus reset.

2. **Bus Reset**

   The pull-up resistor on D– notifies the hub that a low speed (1.5 Mbps) device has just been connected. The host recognizes the presence of a new USB device and initiates a bus reset to that device.

3. **Enumeration**

   The host initiates SETUP transactions that reveal general and device specific information about the mouse. When the description is received, the host assigns a new and unique USB address to the mouse. The mouse begins responding to communication with the newly assigned address, while the host continues to ask for information about the device description, configuration description and HID report description. Using the information returned from the mouse, the host now knows the number of data endpoints supported by the mouse. At this point, the process of enumeration is completed.

4. **Post Enumeration Operation**

   Once communication between the host and mouse is established, the peripheral now has the task of sending and receiving data on the control and data endpoints. In this case, when the host configures endpoint 1, the mouse starts to transmit button and motion data back to the host when there is data to send. At any time the peripheral may be reset or reconfigured by the host.

USB Requests – Endpoint 0

Endpoint 0 acts as the control endpoint for the host. On power-up endpoint 0 is the default communication channel for all USB devices. The host initiates Control- Read and Control-Write (see Chapter 8 of the USB specification) to determine the device type and how to configure communications with the device. In this particular design, only Control-Read transactions are required to enumerate a mouse. For a list of valid requests see Chapter 9 of the USB specification. In addition to the standard "Chapter 9" requests, a mouse must also support all valid HID class requests for a mouse.

USB Requests – Endpoint 1

Endpoint 1 is the data transfer communications channel for mouse button, wheel, and movement information. Requests to this endpoint are not recognized until the host configures endpoint 1. Once this endpoint is enabled, then interrupt IN requests are sent from the host to the mouse to gather mouse data. When the mouse is left idle (i.e. no movement, no new button presses, no wheel movement) the firmware will NAK requests to this endpoint. Data is only reported when there is a status change with the mouse.

Two HID report formats are used in this design. The boot protocol, as defined by the HID specification, is the default report protocol that all USB enabled systems understands. The boot protocol has a three-byte format, and so does not report wheel information. The HID report descriptor defines the report protocol format. This format is four bytes and is the same as the report format with the exception of the fourth byte, which is the wheel information. Appendix F of this document lists the USB Data Reporting Format.

Mouse Firmware Details

The following are descriptions of the functions in HP_SENSOR.ASM.

MainStart – Executed when the mouse is first plugged into the PC. The only interface determined in the firmware is USB.

USBMainLoop – This function spins in an infinite loop waiting for an USB event that needs servicing. The ButtonCheck, WheelCheck and GetMouseDeltaXY functions are called within this loop to retrieve any new motion or button information. The DetermineMouseChange is called to determine if any mouse change for responding USB IN packet. The data received from these functions will be loaded into the endpoint 1 buffer to be sent to the host.

InitializeMouseSensor – This routine resets the serial interface and the ADND-5050 internal registers by generating a pulse on the NRESET pin.

WheelCheck –This routine is called within the infinite USBMainLoop to detect mouse wheel for Z-axis change. This routine compares the current state of the wheel with its last state to detect any changes in the status. If the status change of the wheel remains until the expiration of debounce timer, the new wheel state is confirmed. This routine will record the new wheel state in the DeltaZCounter variable which will be reported to the host in the main loop.

ButtonCheck –This routine is called within the infinite USBMainLoop to detect mouse buttons changes since last check. This routine compares the current state of the buttons with its last state to detect any changes in the status. If the status change of the wheel remains until the expiration of debounce timer (15ms), the new wheel state is confirmed. This routine will record the new button state for right, middle and left buttons in the ButtonStatus variable which will be reported to the host in the main loop. CPI+ and CPI- button status change will increment or decrement the sensor resolution in 125cpi step whenever applicable.

UpdateDPIValue – This routine reads the current sensor resolution in Mouse_Control2 register and then write the register with the latest resolution setting which is stored in DPIValue.

MoveMouseData – This routine move variable contents into USB data buffer prior to sending them to the host. For example of 12-bit X/Y report structure, mouse buttons event is moved to fifo_out1, DeltaXCounterL is moved to fifo_out2, lower-nibble of DeltaXCounterH is moved to lower-nibble of fifo_out3, lower-nibble of DeltaYCounterL is moved to higher-nibble of fifo_out3, higher-nibble of DeltaYCounterL is moved to lower-nibble of fifo_out4, lower-nibble of DeltaYCounterH is moved to higher-nibble of fifo_out4, DeltaZCounter is moved to fifo_out5.

GetMouseDeltaXY – Reads the ADNS-5050 Motion register. The data returned from this register will be used to determine if any motion has occurred. If motion occurs, it reads the ADNS-5050 Delta_X register for the X movement. Calls the ReadHPSensor routine to enable the SPI interface and perform reading operations through the two wire serial interface. Any new X motion information is added to the [DeltaXCounterL] and [DeltaXCounterH] variables. ADNS-5050 Delta_Y register will then be read for the Y movement. Calls the ReadHPSensor routine to enable the SPI interface and perform reading operations through the two wire serial interface. Any new Y motion information is added to the [DeltaYCounterL] and [DeltaYCounterH] variables.

WriteHPSensor – Writes to the ADNS-5050 register. A write operation consists of two bytes. The first byte contains the address (7 bits) and has "1" as its MSB. The second byte contains data. Pin 7 of HT82M99E is configured in resistive mode, allowing the microcontroller to drive both SCLK and SDATA lines. WriteDataByteToSensor is called to carry the write operation.

ReadHPSensor – Reads the desired ADNS-5050 registers. A read operation is composed of two parts. First, the microcontroller performs a write to the ADNS-5050, sending the address of the target register to be read. WriteDataByteToSensor is called to carry the write operation.

WriteDataByteToSensor – Writes the data to be transmitted onto the SPI pins.

SetHPSensorConfiguration – To customise features of sensor, for example the sensor resolution.

CheckHPSensorProductID – This function checks the product ID of the sensor chip being used. The ID returned should match with the ADNS-5050's ID.

USB Firmware Description

A function call map for USB operation is shown in Figure 5. USB related firmware listed as below:

STD1.ASM code processes standard USB request.

StandardRequest – Decode standard USB request and jump to correct subroutine.

SetConfiguration –This routine is entered when a SET CONFIGURATION request has been received from the host.

GetConfiguration –This routine is entered whenever a GET CONFIGURATION Request is received. This function then starts a control read transaction that sends the configuration, interface, endpoint, and HID descriptors to the host.

GetStatus – This routine is entered whenever a GET STATUS request is received. This routine is to respond device status(like remote wakeup and self-power), interface status, or endpoint status(halt or not) to host.

SetClearFeature –This routine is entered whenever a SET FEATURE or CLEAR FEATURE request has been received to set/clear device related feature (like remote wakeup feature) or endpoint related feature (like stall) by host.

SetAddressC –This routine is entered whenever a SET ADDRESS request has been received. The device address change cannot actually take place until after the status stage of this no-data control transaction, and accomplished by hardware circuit.

GetDescriptor –This routine is entered when a GET DESCRIPTOR request is received from the host. This function decodes the descriptor request and sends the proper descriptor.

ToStallPipe0 – Unsupported or invalid descriptor requests will cause this firmware to STALL these transactions.

CLS1.ASM processes USB HID class request.

GetReport –This routine is entered whenever a GET REPORT request is received.

GetSetProtocol –This routine is entered whenever a GET PROTOCOL or SET PROTOCOL request is received. This no-data control transaction enables boot or report protocol.

ToStallPipe0 – Unsupported or invalid descriptor requests will cause this firmware to STALL these transactions.

HT82M99E FIFO access related firmware listed as below:

FIFO0_Rd_Check – Check if FIFO0 is available for read operation.

FIFO0_Wr_Check – Check if FIFO0 is available for write operation.

FIFO1_Wr_check – Check if FIFO1 is available for write operation.

ReadFIFO0 – Read FIFO0 operation.

WriteFIFO0 – Write FIFO0 operation.

WriteFIFO1 – Write FIFO1 operaton.

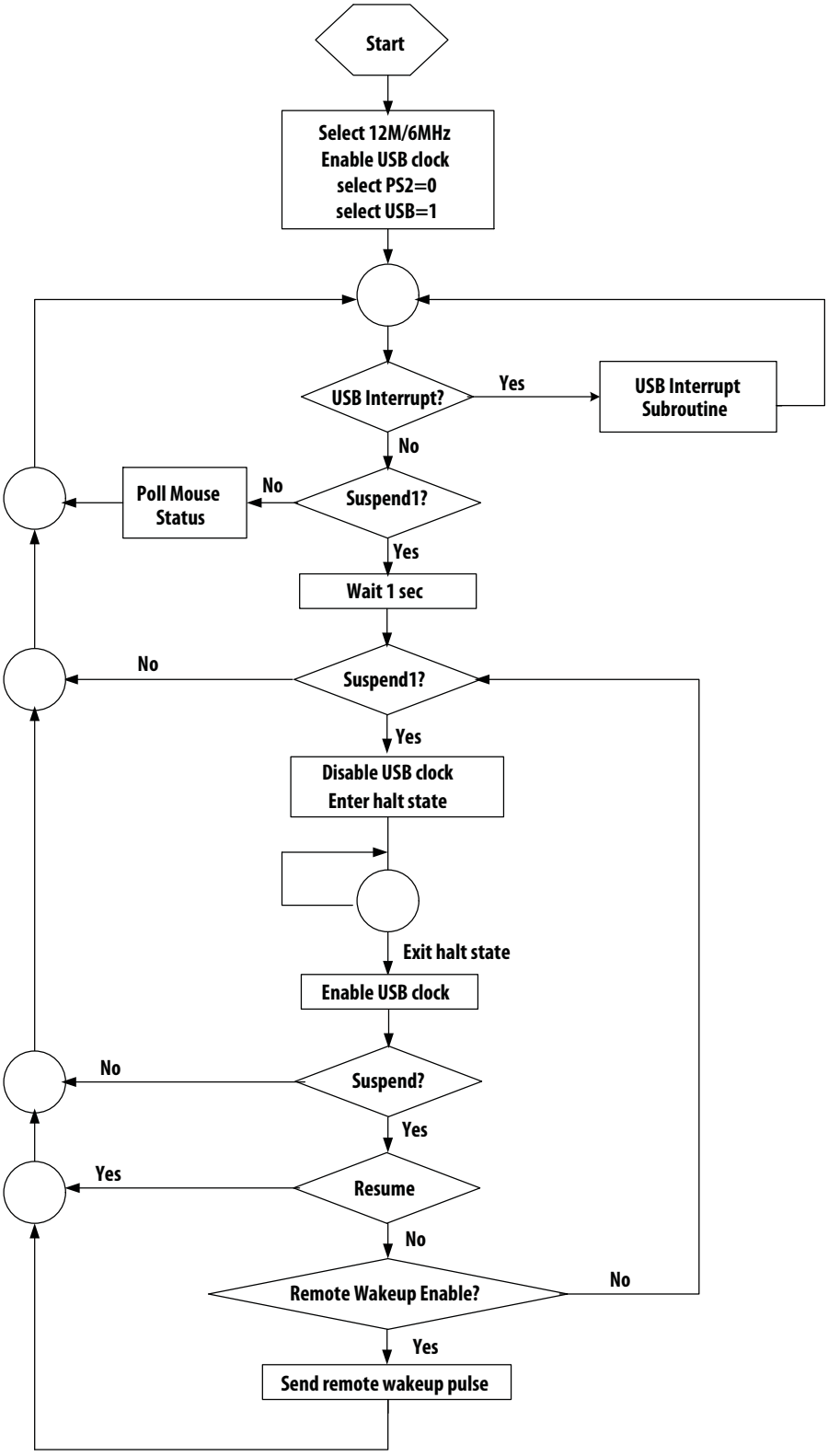


Figure 5. USB Mouse Flow Chart

Interrupt Service Routines (ISR)

The HT82M99E features 2 different sources of interrupts. There is only one ISR, which is USB INT, implemented in this application. The USB interrupts are triggered by the following USB events and the related interrupt request flag (USBF; bit 4 of the INTC) will be set.

- Access of the corresponding USB FIFO from PC
- The USB suspend signal from PC
- The USB resume signal from PC
- USB Reset signal

If an interrupt is enabled and the conditions for the interrupts are met, the microcontroller will generate an interrupt. Upon servicing the interrupt, the hardware will first disable all interrupts by clearing the Global Interrupt Enable bit. This is followed by an automatic CALL instruction to the ROM address of the interrupt being serviced in the Interrupt Vector. The instruction in the Interrupt Vector is typically a JMP instruction to the Interrupt Service Routine (ISR). A RETI or RET instruction at the end of the ISR brings the program counter (PC) back to the location prior to the interrupt (POR and USB Bus Reset are exceptions).

Ext_Start –This is the USB ISR entry point and is entered whenever a USB request happened. It backups TBLP, MP0, MP1, STATUS, ACC registers for this ISR. Check and process resume, suspend signal and fifo access operation.

Exit_of_Ext_Start – This routine is the exit of USB ISR. It restores registers backup in Ext_Start and RETI.

CheckAccessFIF0 – Determine access FIFO0 or FIFO1 and jump to correct procedure.

AccessFIF00 – Process SETUP command, IN token, and OUT token for control pipe. Decode and respond USB stardard or class request to host.

AccessFIF01 – Write mouse motion data including button, X, Y, Z-Wheel to host if any mouse motion occurs.

Manufacturer String*1

A request for the manufacturer string will return the following string.

"Avago Ref Mouse"

Product String*2

A request for the product string will return the following string.

"ADNS-5050 Mouse"

Configuration String

A request for the configuration string will return the following string.

"Avago HID-Compliant Mouse"

Notes:
1. The Manufacturer String should be changed to the name of your company.
2. The Product String should be changed to your product's name.

Appendix A. Schematic Diagram of the Overall Circuit

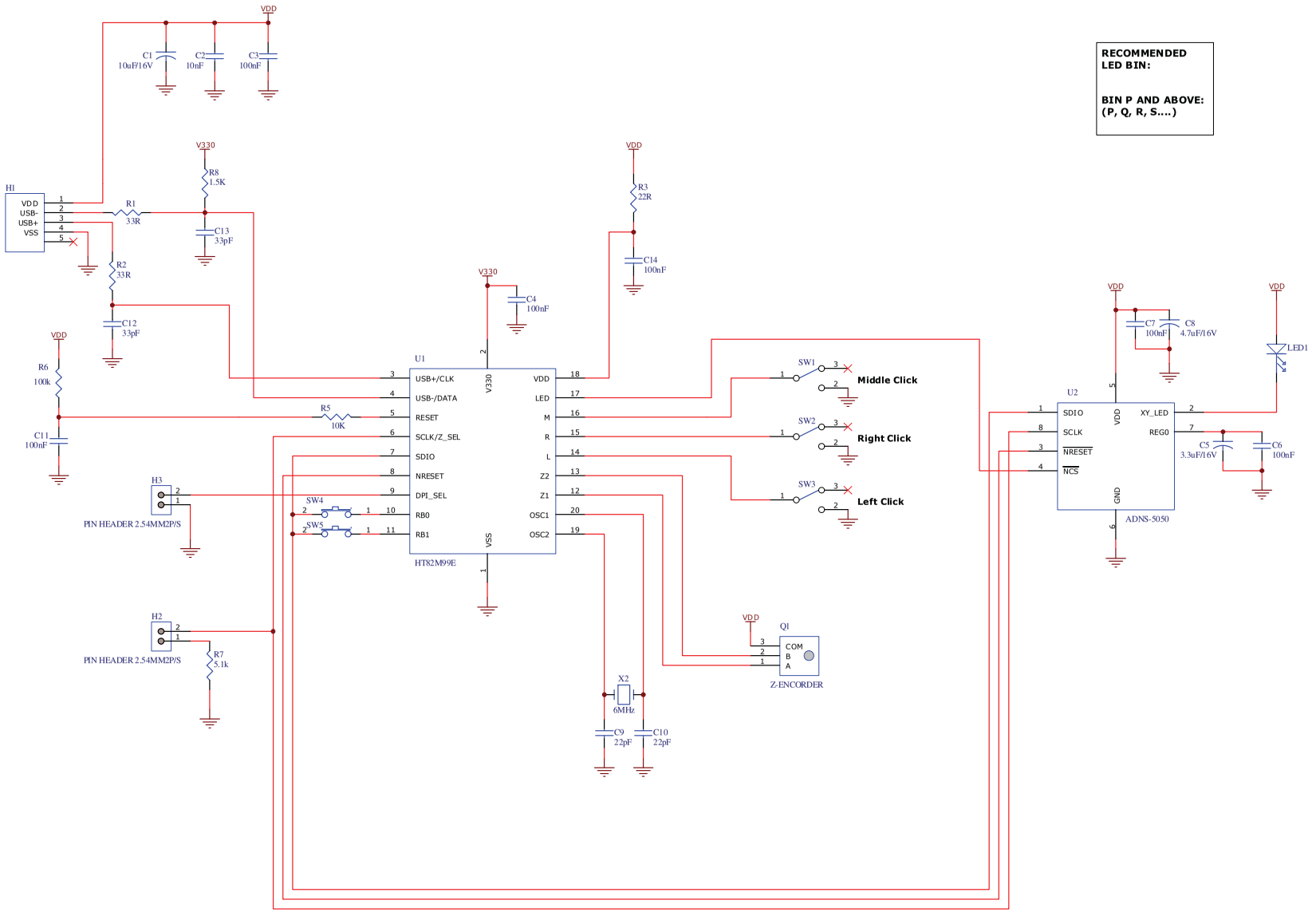


Figure A1. Circuit-level block diagram for ADNK-5053-HT01 designer's kit optical mouse using the Avago ADNS-5050 optical mouse sensor and Holtek HT82M99E USB Controller.

Appendix B. Bill of Materials for Components Shown on schematic

No	Components	Footprint	Designator	Qty
01.	Resistor 1.5K 1% 0.125watt	0603 [SMD]	R8	1
02.	Resistor 100K 1% 0.125watt	0603 [SMD]	R6	1
03.	Resistor 10K 1% 0.125watt	0603 [SMD]	R5	1
04.	Resistor 22R 1% 0.125watt	0805 [SMD]	R3	1
05.	Resistor 33R 1% 0.125watt	0603 [SMD]	R1.R2	2
06.	Resistor 5.1k 1% 0.125watt	0603 [SMD]	R7	1
07.	Ceramic Cap. 10nF 25V	0603 [SMD]	C2	1
08.	Ceramic Cap. 100nF 50V	0603 [SMD]	C3.C4.C6.C7.C11.C14	6
09.	Ceramic Cap. 10uF 16V	0805 [SMD]	C1	1
10.	Ceramic Cap. 22pF 50V	0603 [SMD]	C9.C10	2
11.	Tantalum Cap. 3.3uF,16V	Cap 0.1/POL	C5	1
12.	Ceramic Cap . 33 pF 50V	0603 [SMD]	C12.C13	2
13.	Ceramic Cap 4.7uF, 10V	1206 [SMD]	C8	1
14.	HLMP-EG3E	LED 5mm R/A	LED1	1
15.	Switch – SPNO Tactile	SPNO Tactile	RB0.RB1	2
16.	Switch – Micro switch	SMD	SW1.SW2.SW3	3
17.	Swtich – Cap	SPNO Tactile Cap	RB0* RB1	2
18.	Pin Header 40P	2mm	H1.H2.H3	1
19.	Connector – IC Socket	DIP-18pin	U1*	1
20.	Crystal 6Mhz	SMD/ DIP	X2	1
21.	Mouse Optical Sensor ADNS-5050	ADNS-5050	U2	1
22.	Micro controller HT82M99E	DIP-20pin	U1	1
23.	Z – ENCODER	ZDET	Q1	1

Appendix C. PCB Layout

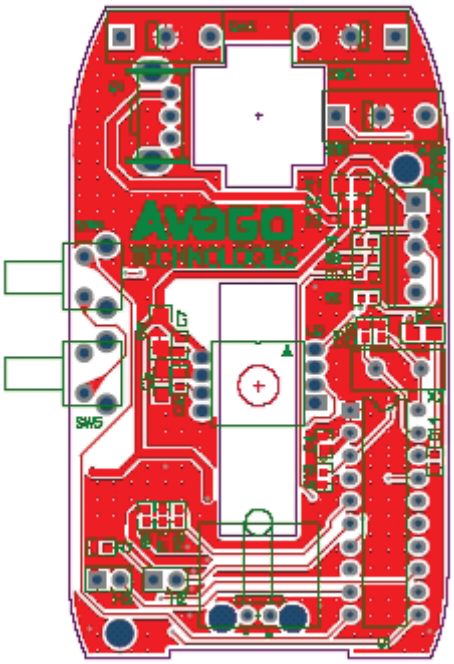


Figure C1. PCB Schematic (Top Layer)

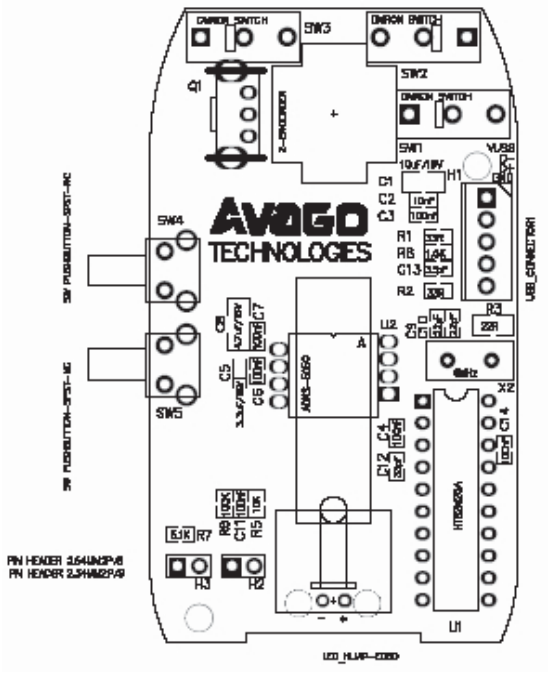


Figure C2. PCB Schematic (Top Overlay)

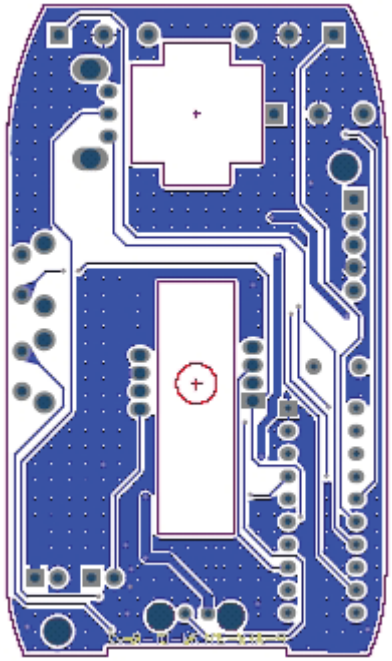


Figure C3. PCB Schematic (Bottom Layer)

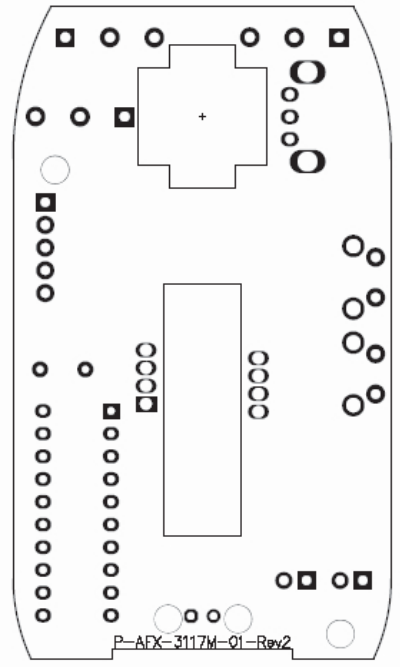


Figure C4. PCB Schematic (Bottom Overlay)

Appendix D. Base Plate Feature

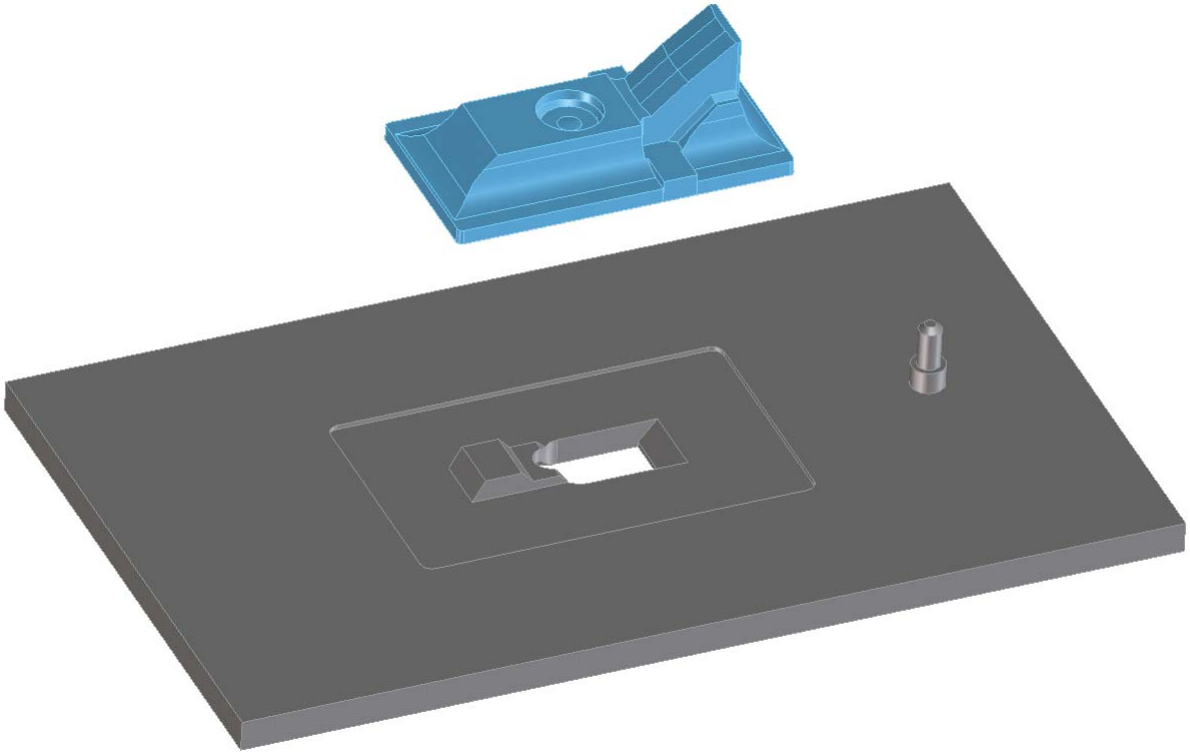


Figure D1. Bottom, top and side view of base plate

Appendix E. Sectional view of PCB assembly

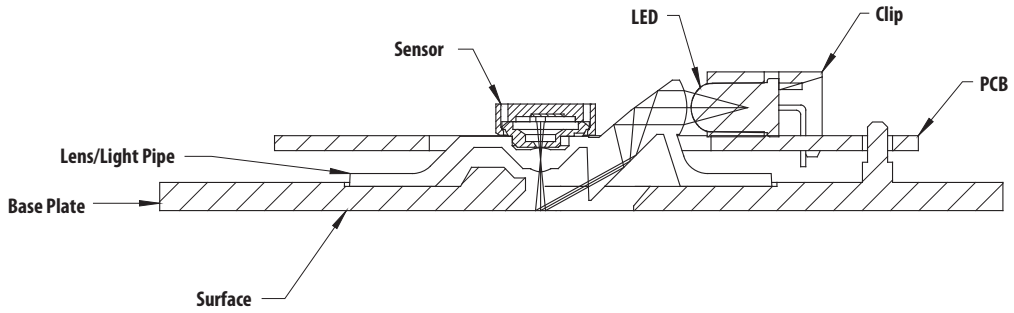


Figure E1. Sectional view of PCB assembly highlighting all optical mouse components (optical mouse sensor, clip, lens, LED, PCB, and base plate).

Appendix F. USB data reporting format

The USB report has two formats, depending on if boot or report protocol is enabled. The following format is the boot protocol and is understood by USB aware BIOS.

	Bit 7							Bit 0
Byte 0	0	0	0	0	0	Middle	Right	Left
Byte 1	X	X	X	X	X	X	X	X
Byte 2	Y	Y	Y	Y	Y	Y	Y	Y

The following is the USB report protocol format and allows the additional wheel movement information in the fourth byte. When the wheel is moved forward the fourth byte reports a 0x01, and when moved backward the fourth byte reports 0xFF. When the wheel is idle, then this byte is assigned 0x00.

	Bit 7							Bit 0
Byte 0	0	0	0	0	0	Middle	Right	Left
Byte 1	X	X	X	X	X	X	X	X
Byte 2	Y	Y	Y	Y	Y	Y	Y	Y
Byte 3	R	R	R	R	R	R	R	F/R

The following is an enhanced USB report protocol format which includes 12-bit of X and Y movements, instead of the conventional 8-bit X and Y movements. Wheel movement information is reported in the fifth byte.

	Bit 7							Bit 0
Byte 0	0	0	0	0	0	Middle	Right	Left
Byte 1	X7	X6	X5	X4	X3	X2	X1	X0
Byte 2	Y3	Y2	Y1	Y0	X11	X10	X9	X8
Byte 3	Y11	Y10	Y9	Y8	Y7	Y6	Y5	Y4
Byte 4	R	R	R	R	R	R	R	F/R

Appendix G. Kit Components

The designer's kit contains components as follows:

Part Number	Description	Name	Quantity
ADNK-5053-HT01 Mouse	Reference Design Mouse	Reference Design Unit	1
ADNS-5050	Solid-State Optical Mouse Sensor	Sensor	5
HT82M99E	Holtek USB Controller	USB Controller	5
ADNS-5100-001	Trim Lens Plate	Lens	5
HLMP-EG3E-xxxxx	Red LED - T1¾ (5 mm) Diameter	LED	5
ADNK-5053-HT01 CD	Includes Documentation and Support Files for ADNK-5053-HT01 **Documentation** a. ADNS-5050 Optical Mouse Sensor Data Sheet. b. HT82M99E Holtek USB Controller Data Sheet. c. ADNS-5100-001 Trim Lens Data Sheet d. HLMP-EG3E-xxxxx LED Data Sheet **Hardware Support Files** a. ADNK-5053-HT01 BOM List b. ADNK-5053-HT01 Schematic c. IGES Base Plate Feature File d. Gerber File **Software Support Files** a. Microcontroller Firmware		1

For product information and a complete list of distributors, please go to our web site: **www.avagotech.com**

AV02-1076EN - May 14, 2012