Copyrights

© Cypress Semiconductor Corporation, 2011-2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and/or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress’ product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

PSoC is a registered trademark and PSoC Designer and enCoRe are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Flash Code Protection

Cypress products meet the specifications contained in their particular Cypress PSoC Datasheets. Cypress believes that its family of PSoC products is one of the most secure families of its kind on the market today, regardless of how they are used. There may be methods, unknown to Cypress, that can breach the code protection features. Any of these methods, to our knowledge, would be dishonest and possibly illegal. Neither Cypress nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as ‘unbreakable’. Cypress is willing to work with the customer who is concerned about the integrity of their code. Code protection is constantly evolving. We at Cypress are committed to continuously improving the code protection features of our products.
# Contents

1. Introduction 5  
   1.1 Kit Contents ................................................................. 5  
   1.2 PSoC® Designer™ .......................................................... 5  
   1.3 PSoC Programmer ........................................................... 5  
   1.4 Additional Learning Resources ........................................ 6  
      1.4.1 References ............................................................ 6  
   1.5 Document History .......................................................... 6  
   1.6 Documentation Conventions ........................................... 6  

2. Getting Started 7  
   2.1 Kit Installation .............................................................. 7  
   2.2 PSoC Designer ............................................................... 11  
   2.3 PSoC Programmer .......................................................... 12  
   2.4 Install Hardware ............................................................ 12  

3. Kit Operation 13  
   3.1 Introduction ................................................................. 13  

4. Hardware 15  
   4.1 Introduction ................................................................. 15  
      4.1.1 Functional Description ............................................. 15  
      4.1.2 Power Supply System .............................................. 20  

5. Code Examples 21  
   5.1 CY4623_RDK ............................................................... 21  
      5.1.1 Project Description .................................................. 21  
      5.1.2 Firmware Architecture .......................................... 22  
      5.1.3 Device Configurations ........................................... 24  
   5.2 Firmware Implementation Details ................................. 28  
      5.2.1 USB and HID Descriptor Modifications .................... 28  
      5.2.2 Mouse Platform Portability .................................... 28  
      5.2.3 Automatic Bus Detection ....................................... 28  
      5.2.4 Initialization ......................................................... 28  
      5.2.5 USB Suspend ......................................................... 28  
      5.2.6 USB Remote Wakeup ............................................. 29  
      5.2.7 Configuration Options .......................................... 30  
   5.3 Code Modules .............................................................. 31  
      5.3.1 APPCONFIG ......................................................... 31  
      5.3.2 CYPDEF .............................................................. 32  
      5.3.3 Mouse ................................................................. 33  
      5.3.4 Optical ............................................................... 35  
      5.3.5 Buttons ............................................................... 37
5.3.6 Wheel ............................................................................................................. 39
5.3.7 PS2_Mouse .................................................................................................... 40
5.3.8 USB_Mouse .................................................................................................. 41
5.3.9 PDC9198 ...................................................................................................... 42
5.3.10 PSOCCONFIGTABLE ................................................................................... 42
5.3.11 ISR ............................................................................................................... 42
5.3.12 PORT ........................................................................................................... 43
5.3.13 Timer ........................................................................................................... 45
5.4 Verify Output ..................................................................................................... 46

A. Appendix .............................................................................................................. 47
A.1 Schematic .......................................................................................................... 47
A.2 Board Layout ..................................................................................................... 48
A.3 Bill of Materials ................................................................................................. 49
1. Introduction

Thank you for your interest in the CY4623 Mouse Reference Design Kit (RDK). This kit offers a complete production-ready solution for a USB or PS/2 optical mouse. The design showcases the revolutionary new enCoRe™ II device family for an overall reduction in system cost.

Cypress offers a complete design for a USB or PS2 mouse in a cost-effective, space-saving solution. This reference design is a fully functional product that complies with all applicable USB, PS/2, and Windows Hardware Quality Labs (WHQL) specifications. Getting Started chapter on page 7 describes the installation and configuration of the CY4623 Mouse RDK. The Kit Operation chapter on page 13 and Hardware chapter on page 15 describe the kit and hardware operations. Code Examples chapter on page 21 demonstrates the code examples provided along with the kit. The Appendix on page 47 provides the schematics and bill of materials (BOM) associated with the CY4623 Mouse RDK.

1.1 Kit Contents

The CY4623 Mouse Reference Design kit includes:

- Evaluation three-button combination optical mouse
- PS/2 adapter
- MiniProg programmer
- USB receptacle programming adapter
- Kit CD/DVD with:
  - Firmware source and object code
  - Complete hardware design files
  - Comprehensive design documentation

Inspect the contents of the kit. If any parts are missing, contact your nearest Cypress sales office for further assistance.

1.2 PSoC® Designer™

PSoC Designer is the integrated design environment (IDE) that you can use to customize your PSoC application. More information about PSoC Designer is available in the PSoC Designer IDE Guide at <Install_Directory>:\Cypress\PSoC Designer\<version>\Documentation.

1.3 PSoC Programmer

PSoC Programmer offers the user a simple GUI that connects to programming hardware to program and configure PSoC devices.
1.4 Additional Learning Resources

Visit http://www.cypress.com for additional learning resources in the form of datasheets, technical reference manual, and application notes.

1.4.1 References

- CY7C63310, CY7C638xx - enCoRe™ II Low Speed USB Peripheral Controller (data sheet)  
http://www.cypress.com/?rID=14212
The ADNS-2620 is a new entry level, small form factor optical mouse sensor
- MiniProg – http://www.cypress.com/?rid=37459  
MiniProg User Guide and Example Projects: This document describes the features and use of the MiniProg device
- PSoC Designer Training – http://www.cypress.com/?rID=40543  
This is a web-based course that provides an overview of the PSoC programmable system-on-chip and its design tools.

1.5 Document History

<table>
<thead>
<tr>
<th>Revision</th>
<th>PDF Creation Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>05/03/2011</td>
<td>CSAI</td>
<td>Initial version of kit guide</td>
</tr>
<tr>
<td>*A</td>
<td>03/28/2012</td>
<td>ELIN</td>
<td>Updated images in Getting Started chapter. Format and content edits throughout the document.</td>
</tr>
</tbody>
</table>

1.6 Documentation Conventions

Table 1-1. Document Conventions for Guides

<table>
<thead>
<tr>
<th>Convention</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier New</td>
<td>Displays file locations, user entered text, and source code: C:\ ...cd\icc\</td>
</tr>
<tr>
<td><em>Italics</em></td>
<td>Displays file names and reference documentation: Read about the sourcefile.hex file in the PSoC Designer User Guide.</td>
</tr>
<tr>
<td>[Bracketed, Bold]</td>
<td>Displays keyboard commands in procedures: [Enter] or [Ctrl] [C]</td>
</tr>
<tr>
<td>File &gt; Open</td>
<td>Represents menu paths: File &gt; Open &gt; New Project</td>
</tr>
<tr>
<td><strong>Bold</strong></td>
<td>Displays commands, menu paths, and icon names in procedures: Click the File icon and then click Open.</td>
</tr>
<tr>
<td>Times New Roman</td>
<td>Displays an equation: 2 + 2 = 4</td>
</tr>
<tr>
<td>Text in gray boxes</td>
<td>Describes cautions or unique functionality of the product.</td>
</tr>
</tbody>
</table>
This chapter describes the installation and configuration of the CY4623 Mouse RDK.

2.1 Kit Installation

To install the kit software, follow these steps:
1. Insert the kit CD/DVD in the CD/DVD drive of your PC. The CD/DVD is designed to auto run and the kit installer startup screen appears. You can also download the latest kit installer from http://www.cypress.com/go/CY4623. Download the installer ISO file and create an installer CD/DVD or extract the ISO using WinRar and install the executables.
2. Click Install CY4623 Mouse RDK to start the installation, as shown in Figure 2-1.

![Figure 2-1. Kit Installer Startup Screen](image)

**Note** If auto run does not execute, double-click the cyautorun.exe file on the root directory of the CD/DVD, as shown in Figure 2-2.
3. The InstallShield Wizard screen appears. The default location for setup is shown on the InstallShield Wizard screen. You can change the location using **Change**, as shown in Figure 2-3.

4. Click **Next** to launch the kit installer.

5. On the **Product Installation Overview** screen, select the installation type that best suits your requirement. The drop-down menu has three options – **Typical**, **Complete**, and **Custom**; see Figure 2-4.

6. Click **Next** to start the installation.
7. After the installation begins, a list of all packages appears on the Installation Page.
8. A green checkmark appears adjacent to every package that is downloaded and installed, as shown in Figure 2-5.
9. Wait until all the packages are downloaded and installed successfully.

Figure 2-5. Installation Page
10. Click **Finish** to complete the installation, as shown in Figure 2-6.

Figure 2-6. Installation Completion Page

![Installation Completion Page](image)

**Note** Following the software installation, verify your installation and setup.

Advanced users can go to the Code Examples chapter on page 21.
2.2 **PSoC Designer**

1. Go to **Start > All Programs > Cypress > PSoC Designer <version>** > PSoC Designer <version>.

2. Click **File > New Project** to create a new project on the PSoC Designer menu; click **File > Open Project** to work with an existing project.

**Figure 2-7. PSoC Designer Interconnect View**

3. To experiment with the code examples, go to **Code Examples chapter on page 21.**

**Note** For more details on PSoC Designer, refer to the PSoC Designer IDE Guide at:

<Install_Directory>:\Cypress\PSoC Designer\<version>\Documentation

The PSoC Designer quick start guide is available at: [http://www.cypress.com/?rID=47954](http://www.cypress.com/?rID=47954).
2.3 PSoC Programmer

1. Click Start > All Programs > Cypress > PSoC Programmer <version>> PSoC Programmer <version>.
2. Select the MiniProg from the port selection, as shown in Figure 2-8.

Figure 2-8. PSoC Programmer Window

3. Click the File Load from the Programmer menu bar, navigate and select the hex file.
4. Use the Program button to program the hex file onto the chip.
5. When programming is successful, Programming Succeeded appears in the Actions pane.

**Note** For more details on PSoC Programmer, see user guide at: 
<Install_Directory>:\Cypress\Programmer\<version>\Documents

2.4 Install Hardware

No hardware installation is required for this kit.
3. Kit Operation

3.1 Introduction

The CY4623 Mouse RDK is a three-button optical mouse with a scroll wheel. The mouse uses an Agilent 2620 optical sensor.

This design uses a single cable that can be connected through a USB connector or PS/2 connector with an adapter. Firmware or code examples can be built to demonstrate USB or PS/2 operation; the chip supports a complete USB-PS/2 combination implementation. The flash-based microcontroller allows easy firmware or code example modification, as well as storage of Vendor and Product IDs without an external EEPROM. The flash can be reprogrammed directly through the USB cable; this makes it possible to update the firmware/code example in manufacturing or in the field without even opening the plastics. The mouse RDK is programmed through the USB connector using a Cypress USB adapter board CY3655-PLG.

Figure 3-1. Cypress USB Programming Adapter

Connect the CY4623 Mouse RDK to the USB adapter board; connect the MiniProg to the ISSP header on the USB adapter, as shown in Figure 3-2. Connect the MiniProg to the PC through A to the Mini B USB cable. On the USB adapter, place two jumpers between pins 1-3 and pins 2-4.

To program the hex file onto the mouse using MiniProg, open PSoC Programmer and select the MiniProg from the Port Selection window. Click Program. While programming is in progress, the Target Power LED on the MiniProg lights up.
Figure 3-2. Programming the Mouse

Detach the MiniProg when **Programming Succeeded** appears in the Actions pane.

Connect the mouse to the PC through the USB connector. Wait for the mouse to be enumerated as a HID mouse. To check the device enumeration, right-click **My Computer > Manage > Device Manager** and start using the mouse.

The mouse works smoothly with the PC. The LED in the mouse illuminates when the mouse is connected to the PC. Check the mouse functionality by moving the cursor in a circular shape, horizontally and vertically; check the functionality of the left, right, and center buttons and the scroll operation.

To change to PS/2 from USB, define the configuration value for **ADD_SUPPORT_FOR_PS2** in the **appconfig.h** file. To build the code and program the hex file, see **PSoC Programmer on page 12**.
4. Hardware

4.1 Introduction

The optical mouse reference design is based on the Cypress enCoRe II – enhanced component reduction – device family. This revolutionary family integrates numerous common components, including the breakthrough crystalless oscillator, 3.3 V regulator with external supply, D– pull-up resistor, and flash memory. The result is an overall reduction in board components and a reduced system cost.

Figure 4-1. Mouse Block Diagram

4.1.1 Functional Description

The mouse board includes an enCoRe II CY7C63813 chip, optical sensor, encoder, three buttons, and a scroll wheel. Figure 4-2 shows the different functional blocks on the CY4623 mouse board.
4.1.1.1 CY7C63813 Chip

The CY7C63813 chip is based on enCoRe II USB, which is a 8-bit microcontroller with Harvard architecture and a M8C CPU speed of up to 24 MHz. It is sourced by an external clock signal. It integrates numerous components including crystalless oscillator, 3.3-V regulator with external supply, D- pull-up resistor, flash memory, and configurable I/O for real world interface without external components. The chip is USB 2.0-USB-IF certified (TID # 40000085).

This chip conforms to the following USB specifications:
- USB Specification, Version 2.0
- USB HID Specification, Version 1.1
- Supports one low speed USB device address
- Supports one control endpoint and two data endpoints
- Integrated USB transceiver with dedicated 3.3 V regulator for USB signaling and D- pull-up.

The CY7C63813 is targeted for the following applications:
- PC HID devices
  - Mouse applications (optomechanical, optical, trackball)
- Gaming
  - Joysticks
  - Game pad
- General purpose
  - Barcode scanners
  - POS terminal
  - Consumer electronics
  - Toys
  - Remote controls
  - Security dongles
Figure 4-3. Schematic View of CY7C63813 Chip
4.1.1.2 Optical Sensor

The ADNS-2620 is a new entry level, small form-factor optical mouse sensor. It is used to implement a nonmechanical tracking engine for computer mouse applications. Unlike its predecessor, this new optical mouse sensor allows for more compact and affordable optical mouse application designs.

The ADNS-2620 is based on the optical navigation technology. It contains an image acquisition system (IAS), a digital signal processor (DSP), and a two-wire serial port. The IAS acquires microscopic surface images through the lens and illumination system provided by the HDNS-2100, HDNS-2200, and HLMP-ED80-xx000. These images are processed by the DSP to determine the direction and distance of motion.

Applications

- Mouse applications for desktop PCs, workstations, and portable PC’s
- Trackballs
- Integrated input devices

Table 4-1. Pin Description for CY7C63813 Chip

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SSEL/P1.3</td>
<td>GPIO Port 1 bit 3. Configure individually. Alternate function is SSEL signal of the SPI bus TTL voltage thresholds.</td>
</tr>
<tr>
<td>2</td>
<td>SCLK/P1.4</td>
<td>GPIO Port 1 bit 4. Configured individually. Alternate function is SCLK signal of the SPI bus TTL voltage thresholds.</td>
</tr>
<tr>
<td>3</td>
<td>SMOSI/P1.5</td>
<td>GPIO Port 1 bit 4. Configured individually. Alternate function is SMOSI signal of the SPI bus TTL voltage thresholds.</td>
</tr>
<tr>
<td>4</td>
<td>SMISO/P1.6</td>
<td>GPIO Port 1 bit 4. Configured individually. Alternate function is SMISO signal of the SPI bus TTL voltage thresholds.</td>
</tr>
<tr>
<td>5</td>
<td>P1.7</td>
<td>GPIO Port 1 bit 7. Configure individually. TTL voltage threshold.</td>
</tr>
<tr>
<td>6</td>
<td>P0.7</td>
<td>GPIO Port 0 bit 7. Configured individually.</td>
</tr>
<tr>
<td>7</td>
<td>P0.6/TIO1</td>
<td>GPIO Port 0 bit 6. Configure individually. Alternate function Timer capture inputs or Timer output TIO1.</td>
</tr>
<tr>
<td>8</td>
<td>P0.5/TIO0</td>
<td>GPIO Port 0 bit 5. Configure individually. Alternate function Timer capture inputs or Timer output TIO0.</td>
</tr>
<tr>
<td>9</td>
<td>P0.4/INT2</td>
<td>GPIO Port 0 bit 4. Configure individually. Optional rising edge interrupt INT2.</td>
</tr>
<tr>
<td>10</td>
<td>P0.3/INT1</td>
<td>GPIO Port 0 bit 3. Configure individually. Optional rising edge interrupt INT1.</td>
</tr>
<tr>
<td>11</td>
<td>P0.2/INT0</td>
<td>GPIO Port 0 bit 2. Configure individually. Optional rising edge interrupt INT0.</td>
</tr>
<tr>
<td>12</td>
<td>P0.1</td>
<td>GPIO Port 0 bit 1. Configured individually. External clock input when configured as Clock Out.</td>
</tr>
<tr>
<td>13</td>
<td>P0.0</td>
<td>GPIO Port 0 bit 0. Configured individually. Clock output when configured as Clock In.</td>
</tr>
<tr>
<td>14</td>
<td>VSS</td>
<td>Ground</td>
</tr>
<tr>
<td>15</td>
<td>D+/P1.0</td>
<td>GPIO Port 1 bit 0/USB D+ - If this pin is used as a General Purpose output, it draws current. This pin must be configured as an input to reduce current draw.</td>
</tr>
<tr>
<td>16</td>
<td>D-/P1.1</td>
<td>GPIO Port 1 bit 1/USB D- - If this pin is used as a General Purpose output, it draws current. This pin must be configured as an input to reduce current draw.</td>
</tr>
<tr>
<td>17</td>
<td>VDD</td>
<td>Supply</td>
</tr>
<tr>
<td>18</td>
<td>VREG/P1.2</td>
<td>GPIO Port 1 bit 2. Configured individually. 3.3 V if regulator is enabled. A1 uF min, 2 uF max capacitor is required on Vreg output.</td>
</tr>
</tbody>
</table>
4.1.1.3 Wheel and Buttons

There are three buttons S1, S2, and S3 for left, right, and middle button operations of the mouse. The wheel with the encoder is used for scroll operation.

Table 4-2. Pin Details of ADNS-2620 Chip

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OSC_IN</td>
<td>Oscillator input</td>
</tr>
<tr>
<td>2</td>
<td>OSC_OUT</td>
<td>Oscillator output</td>
</tr>
<tr>
<td>3</td>
<td>SDIO</td>
<td>Serial Port Data (input and output)</td>
</tr>
<tr>
<td>4</td>
<td>SCK</td>
<td>Serial Port Clock (Input)</td>
</tr>
<tr>
<td>5</td>
<td>LED_CNTL</td>
<td>Digital Shutter Signal Out</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
<td>System Ground</td>
</tr>
<tr>
<td>7</td>
<td>VDD</td>
<td>5V DC Input</td>
</tr>
<tr>
<td>8</td>
<td>REFA</td>
<td>Internal reference</td>
</tr>
</tbody>
</table>

Figure 4-5. Schematic View of Buttons and Wheel
4.1.1.4 **USB Header**

The USB header communicates between the PC and the mouse; this also supplies an input voltage of 5 V to power up the mouse. This header is extended with a cable to connect to the PC through a USB A connector. Pins 1 and 2 are for D- and D+, respectively, and pin 3 is the VBUS, the input power source for mouse.

Figure 4-6. Schematic View of USB Header

![USB Header Schematic](image)

4.1.2 **Power Supply System**

The CY4623 Mouse RDK board is powered from the USB header.

Figure 4-7. Power Supply System Structure

![Power Supply System](image)
5. Code Examples

All code examples are available at: <Install_Directory>:\Cypress\CY4623 Mouse RDK\<version>\Firmware\ or at kit CD/DVD:\Firmware\.

5.1 CY4623_RDK

5.1.1 Project Description
This code example demonstrates the various features and functions offered by the optical mouse.

5.1.1.1 Mouse Application
The RDK firmware design includes configuration options to target building for USB or PS/2. Cypress has provided a mechanism for automatic detection of the attached bus. However, the customer can add this functionality and create a combination USB and PS/2 mouse. The other features include:

- Support for an ADNS-2620 optical sensor
- Support for three buttons and a scroll wheel
- Support for USB remote wakeup on any button press, scroll wheel movement, or mouse movement
- Implement debounce for all button presses

5.1.1.2 USB Functionality

- The RDK firmware uses the enCoRe II DVK USB User Module (UM). The USB UM provides the USB functionality required for the RDK.
- The RDK is designed to be attached to any USB host interface, but the firmware must be compiled correctly for the mouse firmware to function with USB support. The USB User Module is designed to handle the majority of the Chapter 9 requirements of the USB 2.0 specification.
- The RDK firmware includes additional code to support USB suspend, resume, and remote wakeup. The USB User Module is designed to handle USB HID device class definition v1.11.
- The RDK firmware is designed to support both USB boot and report protocols. Most of this requirement is handled within the USB User Module. The USB User Module allows customization of both the standard USB descriptors and USB HID report descriptors.
5.1.1.3  **PS/2 Functionality**

The enCoRe II DVK PS2D User Module is included in the project design to provide the PS/2 functionality required for the RDK. The RDK is designed to be attached to any PS/2 interface, but the firmware must be compiled with the PS/2 flag set for the RDK to function as a PS/2 device.

5.1.1.4  **Error Handling and Recovery Requirements**

A watchdog timer is enabled so that any environmental influence that results in a firmware execution anomaly results in a complete reset of the system without user intervention. If the watchdog timer expires, then the enCoRe II controller is reset.

5.1.2  **Firmware Architecture**

The CY4623 optical mouse RDK firmware relies on the following user modules included in the enCoRe II DVK to create a modular framework:

- PS2D User Module
- USB User Module
- SPI User Module
- MSTimer User Module
- Programmable Interval Timer (PIT) Module

Figure 5-1. Firmware Architecture

The block diagram in Figure 5-1 displays the high-level design of the CY4623 optical mouse RDK. The user modules are precreated, pretested functions that are analogous to a specific onchip peripheral, which allows designers to focus on the creation of the application. A user module typically includes the necessary interrupt handling software and the application programming interface (API) routines required for the application to communicate with the user module. After the user module is configured in PSoC Designer, the application files for each user module can be generated for the project.
5.1.2.1 CY4623 Optical Mouse RDK Application

Besides being the glue logic that ties the user modules together, the RDK application also handles the following:

- Initialization and configuration
- Execution of the polling loop
- PS/2 or USB idle logic
- LED display logic
- Polling of button, scroll wheel and optical sensor
- Debounce support for button presses
- USB suspend, resume, and remote wakeup
- Generating and transferring the mouse reports to the host by way of either the USB or PS2D UM

5.1.2.2 USB User Module

The USB UM provides USB interface between the host and the RDK application. The functionality provided by the USB UM includes:

- USB device interface driver
- USB Chapter 9 compliance
- Mouse HID class support
- Setup wizard for USB descriptor generation and modification
- Support for interrupt and control transfer types
- Runtime support for descriptor set selection
- Standard APIs for programmatic control of the USB UM

5.1.2.3 PS2D User Module

This PS2D UM provides the PS/2 interface between the host and the RDK application. The functionality provided by the PS2D UM includes:

- PS/2 mouse device interface
- Standard APIs for programmatic control over the PS2D UM
- Custom APIs for mouse command sets

5.1.2.4 SPIM User Module

This SPIM UM provides the SPI interface between the ADNS-2620 optical sensor and the RDK application. The functionality provided by the SPIM UM includes:

- SPI initialization
- Flow control APIs for programmatic control over the SPI interface
- Custom APIs for interrupt service routine handlers

5.1.2.5 MSTIMER User Module

This MSTIMER UM provides the one millisecond timer ISR routine. The application programming interface allows the application to deal with the module at a higher level. Custom code is added to the interrupt service routine, which can be found in the MSTIMERINT.ASM file.
5.1.2.6  Programmable Interval Timer (PIT) User Module

This PITIMER12 UM provides a programmable timer ISR routine using the low power oscillator (ILO). This timer runs while the enCoRe II chips are in low power sleep mode when the 24 MHz internal clock is not available. The PIT is used to turn off/on the optical sensor while in sleep mode, thus providing low power optimization.

5.1.3  Device Configurations

The enCoRe II is configured using the Device Editor in PSoC Designer. The enCoRe II is configured to use a 6 MHz clock and watchdog is enabled. The project includes the PS/2, USB, SPI, PIT, and MSTIMER User Modules. The UM code generated by the project is placed in the library directory.

Figure 5-2. Device Configuration for CY4623_RDK Project

5.1.3.1  PS2D UM Configuration

PS/2 mouse selected within the PS2D Device Support Option dialog.

User Module Parameters

- PS2D_Port = Port_1
- TxBufferSize = 4

5.1.3.2  USB UM Configuration

Adds appropriate USB and HID descriptors using the USB Setup Wizard dialog.
5.1.3.3 **SPIM UM Configuration**

User Module Parameters

- **InDispatchMode = ActiveStatus**
- **SMOSI Enable (P1.5) = Enabled**
- **SMISO Enable (P1.6) = Disabled**
- **BitOrder = MSBFirst**
- **CPOL = High**
- **CPHA = High**
- **ClockCounter = 96**

5.1.3.4 **MSTIMER UM Customization**

The `_MSTIMER_ISR` function located in `MSTIMERINT.ASM` library file is provided to receive the ISR and allow developers to add custom code. However, the RDK firmware does not use this predefined function. Instead, `_MSTIMER_ISR` is declared in the application's `Timer.h` and defined in `Timer.c`. By defining the `_MSTIMER_ISR` function in the application, this version of the function is linked against by PSoC Designer instead of the function defined in the user module's `MSTIMERINT.ASM` file. This has the following benefits:

- The code is located in the application files instead of file in the `lib` directory, making this function easier to find and less likely to be accidentally deleted by a developer or by automated tools.
- ISR now calls directly into the C code without using a proxy assembly file. Also, the compiler now handles management of the stack.

The following code snippet demonstrates how the `_MSTIMER_ISR` is commonly used, with custom code added to call an application function. The assembly code shown must be added between the custom code banners; otherwise, the changes are overwritten every time the project code is generated. Also, notice the need to add stack management code:

```Assembly
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
; NOTE: interrupt service routines must preserve the values of the A and X CPU registers.
PUSH A ; Save Context
PUSH X
LCALL _isr_ms_timer
POP X ; Restore Context
POP A
;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
```

The declaration of the RDK firmware's `_MSTIMER_ISR` function in the application file `Timer.h` is shown in the code that follows:

```C
#pragma interrupt_handler MSTIMER_ISR;
void MSTIMER_ISR( void );
```
Similar to all ISRs, the code should do as little as possible. Because the compiler handles the stack function calls from the ISR, it can cause the entire stack frame to be saved, thus increasing the code size and execution time. The RDK firmware's MSTIMER_ISR function is listed in the following code snippet and demonstrates how to add inline assembly to keep the compiler from being inefficient and saving off the entire stack frame.

```c
void MSTIMER_ISR( void )
{
    if ( delay_counter )
    {
        delay_counter--;
    }
    //Determine if the usb_activity_timeout has expired
    if ( usb_activity_timeout )
    {
        //Decrement usb_activity_timeout
        usb_activity_timeout--;
        //Get current USB activity state.
        //
        // NOTE: this is a call to an assembly function.
        // Because this is a interrupt handler 'C' function the
        // compiler will push and pop all the virtual registers
        // if a standard 'C' function call is used. Therefore,
        // inline assembly is used to avoid the unneeded push
        // and pops onto the stack
        asm ("push A");
        asm ("push X");
        asm ("LCALL USB_bCheckActivity");
        asm ("pop X");
        asm ("mov [usbActivity], A");
        asm ("pop A");
        //Check for USB activity
        if ( usbActivity )
        {
            //Activity detected, reset suspend timer
            usb_activity_timeout = USB_ACTIVITY_TIMEOUT;
        }
```
else suspend condition (usb_activity_timeout has expired)
else
{
// indicate USB suspend detected
shouldSuspend = TRUE;
}

5.1.3.5 Boot.TPL Customization

PSoC Designer uses the template file BOOT.TPL located in the project directory to generate BOOT.ASM. Any changes made to BOOT.ASM are overwritten every time the project code is generated; therefore, the following changes are made to BOOT.TPL, and not BOOT.ASM directly.

The PORT0_ISR function found in the BOOT.TPL file is modified to set the _shouldSuspend global variable found in the RDK application file mouse.c. The pins on Port_0 are mapped to the mouse buttons. Therefore, enabling the interrupt on Port_0 causes the PORT0_ISR routine to get called when any button is pressed. The modified code is as follows:

;-----------------------------------------------
; Insert your custom code below this banner
;-----------------------------------------------
; NOTE: interrupt service routines must preserve
; the values of the A and X CPU registers.
; Set the shouldSuspend flag to false
AND [_shouldSuspend], 0
;-----------------------------------------------
; Insert your custom code above this banner
;-----------------------------------------------

The custom assembly code in this section is added between the custom code banners in the BOOT.TPL file; otherwise the changes are overwritten every time the project code is generated.
5.2 **Firmware Implementation Details**

5.2.1 **USB and HID Descriptor Modifications**

The USB User Module allows customization of both the standard USB descriptors and USB HID report descriptors using a GUI interface incorporated in the USB Setup Wizard. The USB Setup Wizard is an integrated feature included in the USB User Module.

After modifications to the descriptors using the USB Setup Wizard, the USB and HID descriptors can be updated by regenerating the USB library files again.

5.2.2 **Mouse Platform Portability**

The mouse firmware is designed to be portable from one mouse platform to another by remapping the pins on the enCoRe II. The file `pdc9198.h` defines the mouse port and pin mapping definitions that are used throughout the code.

5.2.3 **Automatic Bus Detection**

The RDK can be built to support only USB or both USB and PS/2 interfaces. The necessary hooks to enable an automatic bus detection mechanism are provided by the RDK. However, it is expected that the mechanism to detect the attached bus, if required, is provided by the customer.

To add automatic bus detection, add the implement to the `detect_if_ps2()` routine found in the `mouse.c` application file. This routine is called by the `mouse_init()` routine, also found in the `mouse.c` application file, to configure the global `is_ps2` flag.

5.2.4 **Initialization**

Initialization of the enCoRe II chip is performed by a code that is generated in `boot.asm` by the PSoC Designer software. The module `boot.asm` calls `main()` after enCoRe II is configured and initialized. Main () initializes the components of the mouse application along with the LED, countdown timers, and starts either the PS/2 or USB UM. The main routine then initiates an infinite loop to monitor mouse activity, send mouse reports, and handle suspend/resume activities as required.

5.2.5 **USB Suspend**

The `usb_activity_timeout` is first initialized to the `USB_ACTIVITY_TIMEOUT` value. During the `MSTIMER_ISR` routine, the `usb_activity_timeout` counter is decremented and USB activity is checked by calling the USB User Module function `USB_bCheckActivity`. If there is any USB activity, the `usb_activity_timeout` counter is reset to the `USB_ACTIVITY_TIMEOUT` value. If `usb_activity_timeout` ever reaches zero, the `shouldSuspend` flag is set to indicate a USB Suspend state is detected. In the `main()` idle routine, if the `shouldSuspend` flag indicates USB suspend is detected, then the following is executed:

- Checks to determine if the host enabled the remote wakeup capability for the device
  - If yes, the button press interrupt is enabled so that any button press wakes up the device from the sleep loop. The PIT timer is calibrated for use while in suspend
  - If no, the button press interrupt is not enabled and no button press wakes the device from the sleep loop
- Disables MS Timer interrupt
- Clears the watchdog timer
- Powers down scroll wheel
- Powers down optical sensor
- Turns off LED
- Sets the OSC_CR0 register for the sleep timer interval
- Enables Sleep interrupt
- Enables PIT timer if remote wakeup is enabled
- Calls USB_Suspend to place the USB transceiver in power down mode
- Enters sleep loop until one of the following conditions are detected:
  - USB activity
  - A button interrupt is triggered
If remote wakeup is enabled, then any scroll wheel movement or mouse movement will exit the sleep loop.
- Puts the USB transceiver back in normal operation
- Applies pull-ups to scroll wheel again
- If sleep loop exits because of a user intervention (such as button press, scroll wheel movement, or mouse movement), then initiate the USB Remote Wakeup procedure as follows:
  - Waits for approximately 5 ms to allow hubs to get into their suspend state and prepare for propagating the resume signal.
  - Forces the K state onto the bus (resume).
  - Holds the resume signal for at least 1 ms, but no longer than approximately 10 ms or until you see USB activity.
  - Stops driving the bus and put its drivers into the high impedance state.
- Resets USB activity counter
- Disables button press interrupt
- Disables Sleep interrupt
- Disables PIT interrupt
- Enables MS Timer interrupts again
- Turns on power to optical sensor
- Turns on scroll wheel
- Turns on LED

5.2.6 USB Remote Wakeup

5.2.6.1 Button Press Interrupt

All of the mouse button pins are attached to GPIO Port 0 and by simply enabling the Port 0 interrupt, any button press triggers the Port 0 interrupt. When enabled, the Port 0 interrupt causes the PORT0_ISR function to get called, which in turn disables the shouldSuspend flag. Refer to BOOT.TPL Customization on page 27 for more details on PORT0_ISR.

5.2.6.2 Scroll Wheel Suspend/Remote Wakeup Operation

The pins for the scroll wheel must be polled to detect movement. Pull-ups must be reapplied to the pins for the scroll wheel, before reading the pins states. The pull-ups need a delay to stabilize before reading the pin states, thus USB_bCheckActivity() is called to provide this delay and detect USB activity.

If the host enabled remote wakeup, the state of the scroll wheel pins are read and compared against the state of the pins prior to suspend. If they differ, the user has moved the scroll wheels and mouse and must set the shouldSuspend flag to false begin the remote wake up operation.
Even if the host has not enabled remote wake up, the scroll wheels need to be checked to ensure the scroll wheel doesn't draw current while in suspend. The state of the scroll wheel pins are read and compared against the state of the pins prior to suspend, if they differ the pull-ups on the pins are reconfigured to ensure the mouse stays in a low power state.

5.2.6.3 Optical Sensor Remote Wakeup Operation

Detecting mouse movement through the optical sensor is straightforward: turn on the optical sensor, wait for it to power on, and then perform a read. However, this process becomes more complex when combined with the low power requirements of USB devices.

The Programmable Interval Timer (PIT) is used in the mouse firmware design specifically to implement the optical sensor remote wakeup operation. The PIT timer increments the sleepTicks variable. This creates four time periods. While the value of the sleepTicks variable is less than the SLEEP_IDLE_TICK_COUNT, the optical sensor is off. After the PIT timer increments the value of the sleepTicks variable to be greater than the SLEEP_IDLE_TICK_COUNT, the optical sensor is powered on. The optical sensor takes 50 ms to wakeup; during this period the firmware sleeps. When the value of the sleepTicks variable exceeds the SLEEP_OPTICAL_TICK_COUNT, the optical sensor is read to detect movement.

If movement is detected, the value of the shouldSuspend flag is set to false to begin the remote wake up operation. Otherwise, the optical sensor is powered off and the sleepTicks variable is reset to zero, and the optical sensor procedure repeats.

5.2.7 Configuration Options

All configuration options for the application can be found in the appconfig.h file.

5.2.7.1 ADD_SUPPORT_FOR_PS2

This configuration definition is used to selectively compile PS/2 support for the USB mouse. If this value is defined, then PS/2 and the bus detection support are compiled into the executable image. Without this definition, a USB only mouse is created.

5.2.7.2 MOUSE_DEBOUNCE_COUNT

The button debounce logic detects changes in button state and immediately indicates a change causing a report to be sent to the host. The debounce logic then blocks out any further button state changes for the specified debounce time. This operation is somewhat different from the usual method of waiting for a button to stabilize, during a debounce time, and then reporting the change in button state. It is implemented this way to improve button-reporting latency.

5.2.7.3 PLATFORM_H

This configuration value identifies the header file that has the platform configuration information. The default value is "pdc9198.h", which contains the configuration for the mouse board that is shipped with the RDK. It is anticipated that this macro changes when the code is ported to the customer's mouse platform.
5.3 Code Modules

This section describes the module contents. For operation concepts, see Firmware Architecture on page 22.

5.3.1 APPCONFIG

The APPCONFIG module is the configuration file for the mouse application. Various features are turned on/off from this file.

5.3.1.1 Include Dependency Graphs

This graph shows which files directly or indirectly includes appconfig.h.

Figure 5-3. appconfig.h Include Graph

5.3.1.2 Defines and Types

Table 5-1. APPCONFIG Module Defines and Types

<table>
<thead>
<tr>
<th>Define Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLATFORM_H</td>
<td>This is the definition for the platform specific header file. Other modules can include this definition and when the platform header file changes, the change must happen in this definition only.</td>
</tr>
<tr>
<td>ADD_SUPPORT_FOR_PS2</td>
<td>This configuration value determines if PS/2 support is complied into the project.</td>
</tr>
<tr>
<td>MOUSE_DEBOUNCE_COUNT</td>
<td>This definition is for the number of poll periods to wait during button debounce.</td>
</tr>
<tr>
<td>ADNS_2620</td>
<td>This definition identifies the optical sensor being used.</td>
</tr>
<tr>
<td>USB_DELAY_COUNT</td>
<td>This delay value is used in usb_idle() to satisfy the device’s polling interval for how often USB updates must be sent.</td>
</tr>
<tr>
<td>REMOTE_WU_OPTICAL</td>
<td>This precomplier definition is used to identify blocks of code that is used for Remote Wakeup using the optical sensor.</td>
</tr>
</tbody>
</table>
5.3.2 CYPDEF

The CYPDEF module is an include file for the mouse application with Cypress standardized typedefs. These typedefs are used throughout the code and included by nearly every module.

5.3.2.1 Include Dependency Graphs

This graph shows which files directly or indirectly includes cypdef.h:

Figure 5-4. cypdef.h Include Graph

5.3.2.2 Defines and Types

<table>
<thead>
<tr>
<th>Define Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>This definition is provided for consistency across compilers.</td>
</tr>
<tr>
<td>Type Definitions</td>
<td>These definitions are used to provide consistency across compilers.</td>
</tr>
<tr>
<td>MIN</td>
<td>This macro function provides a logical comparison of two values and returns the lesser value.</td>
</tr>
<tr>
<td>MAX</td>
<td>This macro function provides a logical comparison of two values and returns the greater value.</td>
</tr>
</tbody>
</table>
5.3.3 Mouse

The mouse module contains the main entry point, a routine for data acquisition from the various components of the mouse, and PS/2 detection.

5.3.3.1 Include Dependency Graphs

This graph shows which files directly or indirectly include mouse.h.

Figure 5-5. mouse.h Include Graph

The following graph shows which files directly or indirectly include dependency graph for mouse.c.
5.3.3.2 Defines and Types

Table 5-3. Mouse Module Defines and Types

<table>
<thead>
<tr>
<th>Define/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_TX_PACKET</td>
<td>This structure defines the data payload portion of the USB or PS/2 transmit packet. The data packets include values for the buttons, zwheel, and optical sensor.</td>
</tr>
<tr>
<td>MOUSE_WAIT_BUTTON_UP</td>
<td>This condition sets whether pressing a button to exit sleep mode is passed to the PC or not by either waiting for the button to go up or not. Its value changes based upon the sleep mode selected in appconfig.h.</td>
</tr>
<tr>
<td>PORT0_UNUSED_PINS</td>
<td>A bit for pin association for all the pins that are not used in the mouse design. All unused pins should be driven to outputs to prevent suspend power issues.</td>
</tr>
<tr>
<td>PORT1_UNUSED_PINS</td>
<td>A bit for pin association for all the pins that are not used in the mouse design. All unused pins should be driven to outputs to prevent suspend power issues.</td>
</tr>
</tbody>
</table>
5.3.3.3 Variable Definitions

Table 5-4. Mouse Module Variable Definitions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_ps2</td>
<td>This variable is a boolean value. It is used to determine if the mouse is connected to a PS/2 bus or a USB.</td>
</tr>
<tr>
<td>report_packet</td>
<td>This variable is used to reference the report packets of type APP_TX_PACKET.</td>
</tr>
<tr>
<td>shouldSuspend</td>
<td>This is a flag to indicate USB Suspend. Default is not in USB Suspend state.</td>
</tr>
<tr>
<td>bUsbActive</td>
<td>This conditional is used as a USB activity detection indicator for USB state. bUSBActive is set to 0 prior to going into suspend.</td>
</tr>
</tbody>
</table>

5.3.3.4 Functions

Table 5-5. Mouse Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>external</td>
<td>This function is the main entry point for the application. It initializes all mouse application components and executes the poll loop.</td>
</tr>
<tr>
<td>mouse_init</td>
<td>static</td>
<td>This function initializes all components of the mouse application.</td>
</tr>
<tr>
<td>detect_if_ps2</td>
<td>internal</td>
<td>This function is unimplemented. It is to be used for the USB / PS/2 detection algorithm.</td>
</tr>
<tr>
<td>mouse_get_report</td>
<td>external</td>
<td>This function polls the optics, wheel and buttons for potential events to send over the radio. It also builds the packet payload.</td>
</tr>
<tr>
<td>mouse_power_up</td>
<td>external</td>
<td>This function performs the power on sequence for the mouse when resuming from sleep mode.</td>
</tr>
<tr>
<td>mouse_power_down</td>
<td>external</td>
<td>This function performs the power down sequence for the mouse before entering sleep mode.</td>
</tr>
<tr>
<td>init_mouse_report</td>
<td>static</td>
<td>This utility function zeros out the report_packet.</td>
</tr>
</tbody>
</table>

5.3.4 Optical

The optical module provides low-level routines for communicating with the optical sensor chip. Standard enCoRe II GPIO pins are used as a serial port and manually manipulated to meet timing requirements to the optical sensor chip. This module provides the interface to read and write registers on the optical sensor chip as well as managing the power saving modes of the optical sensor.

5.3.4.1 Include Dependency Graphs

This graph shows which files directly or indirectly includes optical.h.
5.3.4.2 Defines and Types

Table 5-6. Optical Module Defines and Types

<table>
<thead>
<tr>
<th>Define/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTICAL_REPORT</td>
<td>This structure defines the layout of the X and Y delta data within the transmission packet payload.</td>
</tr>
<tr>
<td>Optical Definitions</td>
<td>These definitions are used to interface with the Agilent optical sensor. They define register addresses and bit masks for those registers.</td>
</tr>
</tbody>
</table>
5.3.4.3 Functions

Table 5-7. Optical Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>optical_init</td>
<td>external</td>
<td>This function resets, establishes communication and initializes the optical sensor.</td>
</tr>
<tr>
<td>optical_get_report</td>
<td>external</td>
<td>This function polls the optical sensor for X/Y motion detail and prepares the report. It also handles overflow conditions.</td>
</tr>
<tr>
<td>optical_power_up</td>
<td>external</td>
<td>This function brings the optical sensor out of a sleep mode.</td>
</tr>
<tr>
<td>optical_power_down</td>
<td>external</td>
<td>This function puts the optical sensor into a sleep mode.</td>
</tr>
<tr>
<td>adns_read</td>
<td>static</td>
<td>This function reads an optical sensor register at a given address.</td>
</tr>
<tr>
<td>adns_write</td>
<td>static</td>
<td>This function writes a value to a given address in the optical sensor.</td>
</tr>
</tbody>
</table>

5.3.5 Buttons

The buttons module provides support for the left, middle, and right button code. Additionally, this module includes button interrupt and debounce code.

5.3.5.1 Include Dependency Graph

This graph shows which files directly or indirectly includes buttons.h:

Figure 5-9. buttons.h Include Graph

This graph shows which files directly or indirectly include dependency graph for button.c.

Figure 5-10. button.c Include Dependency Graph
5.3.5.2 Defines and Types

Table 5-8. Buttons Module Defines and Types

<table>
<thead>
<tr>
<th>Define/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT_BUTTON_REPORT_BIT</td>
<td>Bit position for the left button in the button report.</td>
</tr>
<tr>
<td>RIGHT_BUTTON_REPORT_BIT</td>
<td>Bit position for the right button in the button report.</td>
</tr>
<tr>
<td>MIDDLE_BUTTON_REPORT_BIT</td>
<td>Bit position for the middle button in the button report.</td>
</tr>
<tr>
<td>NUMBER_OF-buttons</td>
<td>Defines the number of buttons implemented on the mouse.</td>
</tr>
<tr>
<td>DEBOUNCE_QUEUE_LEN</td>
<td>Defines the length of the event queue. Defined to be the NUMBER_OF_BUTTONS.</td>
</tr>
<tr>
<td>INVALID_INDEX</td>
<td>Defines a mask for invalid button values.</td>
</tr>
<tr>
<td>BUTTON_REPORT</td>
<td>Defines the button report layout for the transmission data packet.</td>
</tr>
<tr>
<td>DEBOUNCE_ENTRY</td>
<td>Defines the debounce queue entry value and index variables.</td>
</tr>
<tr>
<td>MOUSE_LR_BUTTONS</td>
<td>A simplifying macro used in initialization and selection of the left and</td>
</tr>
<tr>
<td></td>
<td>right buttons.</td>
</tr>
</tbody>
</table>

5.3.5.3 Variable Definitions

Table 5-9. Buttons Module Variable Definitions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_button_state</td>
<td>This variable maintains the state of the buttons when last polled.</td>
</tr>
<tr>
<td>debounce_queue</td>
<td>This array of values is used for debouncing button presses.</td>
</tr>
</tbody>
</table>

5.3.5.4 Functions

Table 5-10. Buttons Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttons_isr</td>
<td>external</td>
<td>This function is an interrupt handler for when a button is pressed. It is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>only activated when the mouse is asleep and is used to wake up the mouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>when a button is pressed.</td>
</tr>
<tr>
<td>buttons_init</td>
<td>external</td>
<td>This function initializes the port, state, and interrupt handler for the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>buttons component.</td>
</tr>
<tr>
<td>buttons_power_up</td>
<td>external</td>
<td>This function performs a power up action for the buttons component when</td>
</tr>
<tr>
<td></td>
<td></td>
<td>coming out of sleep.</td>
</tr>
<tr>
<td>buttons_power_down</td>
<td>external</td>
<td>This function performs the power down action for the buttons component of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enabling interrupts for the buttons, before going to sleep.</td>
</tr>
<tr>
<td>buttons_get_report</td>
<td>external</td>
<td>This function polls the button status and performs debouncing on any keys</td>
</tr>
<tr>
<td></td>
<td></td>
<td>that change state. It also formats the report for transmission.</td>
</tr>
<tr>
<td>buttons_up</td>
<td>external</td>
<td>This helper function returns true if all buttons are up and false if a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>button is still held down.</td>
</tr>
<tr>
<td>get_button_state</td>
<td>static</td>
<td>This helper function reads the state of the buttons and translates button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>state to report state.</td>
</tr>
<tr>
<td>debounce_init</td>
<td>external</td>
<td>This function initializes the debounce logic for the buttons.</td>
</tr>
<tr>
<td>age_debounce_queue</td>
<td>static</td>
<td>This function decrements debounce values for all keys every time the mouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>buttons are polled. This provides a debounce period for when a button is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pressed.</td>
</tr>
</tbody>
</table>
5.3.6 Wheel

This module handles the interrupts generated by the Z wheel and computes a delta that is read at the defined poll rate. The Z wheel is a mechanical switch that can be left in a position that wastes battery current. Because of this, the Z wheel is deactivated in sleep mode and state is lost. However, the button associated with the Z wheel is still active.

5.3.6.1 Include Dependency Graph

This graph shows which files directly or indirectly includes wheel.h.

Figure 5-11. wheel.h Include Graph

This graph shows which files directly or indirectly include dependency graph for wheel.c.

Figure 5-12. wheel.c Include Dependency Graph

5.3.6.2 Defines and Types

Table 5-11. Wheel Module Defines and Types

<table>
<thead>
<tr>
<th>Define/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEEL_REPORT</td>
<td>This structure defines the report for the transmission data payload for the Z wheel.</td>
</tr>
<tr>
<td>WHEEL_STATE</td>
<td>This structure defines Z wheel state data that needs to be persistent.</td>
</tr>
<tr>
<td>MOUSE_WHEEL_STATE0</td>
<td>Bit translation for Z wheel port input.</td>
</tr>
<tr>
<td>MOUSE_WHEEL_STATE1</td>
<td>Bit translation for Z wheel port input.</td>
</tr>
</tbody>
</table>
5.3.6.3 **Variable Definitions**

Table 5-12. Wheel Module Variable Definitions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wheel_state</td>
<td>This is a state variable used to maintain the state of the Z wheel from interrupt to interrupt until the Z delta is polled.</td>
</tr>
<tr>
<td>gZWheelState1</td>
<td>This is a state variable used to capture the state of pin 1 of the zwheel before going into suspend. It is then used in suspend to identify movement of the zwheel.</td>
</tr>
<tr>
<td>gZWheelState2</td>
<td>This is a state variable used to capture the state of pin 2 of the zwheel before going into suspend. It is then used in suspend to identify movement of the zwheel.</td>
</tr>
</tbody>
</table>

5.3.6.4 **Functions**

Table 5-13. Wheel Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wheel_init</td>
<td>External</td>
<td>This function initializes the Z wheel port and state data.</td>
</tr>
<tr>
<td>wheel_get_report</td>
<td>External</td>
<td>This function formats the Z wheel delta data for the report and clears state data.</td>
</tr>
<tr>
<td>wheel_power_up</td>
<td>External</td>
<td>This function performs a power up action for the Z wheel component when the mouse exits sleep mode.</td>
</tr>
<tr>
<td>wheel_power_down</td>
<td>External</td>
<td>This function performs a power down action for the Z wheel before the mouse enters sleep mode. It is required to turn off the drive for the Z wheel when the mouse sleeps to save battery life. The sensor could be left in a power drain position.</td>
</tr>
<tr>
<td>wheel_poll</td>
<td>External</td>
<td>This function looks for transitions to take place on the Z wheel pins. It captures the change and records it in the state variable.</td>
</tr>
<tr>
<td>wheel_get_state</td>
<td>Static</td>
<td>This is a helper function used to read the Z wheel port.</td>
</tr>
</tbody>
</table>

5.3.7 **PS2_Mouse**

This module provides the implementation for communication transactions with the PS2D UM. This module is very simple because the PS2D UM provides most of the functionality.

5.3.7.1 **Include Dependency Graph**

The following figure shows the include dependency graph for ps2_mouse.c.

Figure 5-13. ps2_mouse.c Include Dependency Graph
5.3.7.2 Functions

Table 5-14. PS2_Mouse Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ps2_init</td>
<td>static</td>
<td>This function initializes the PS2D UM and waits for PS/2 to become ready before returning.</td>
</tr>
<tr>
<td>ps2_idle</td>
<td>static</td>
<td>This function implements idle PS/2 mouse functionality. This function is called from the main work loop to detect button, wheel, or mouse movements and to process PS/2 transactions.</td>
</tr>
</tbody>
</table>

5.3.8 USB_Mouse

This module provides the implementation for communication transactions with the USB User Module. This module is very simple because the USB User Module provides most of the functionality.

5.3.8.1 Include Dependency Graph

The following figure shows the include dependency graph for usb_mouse.c:

Figure 5-14. usb_mouse.c Include Dependency Graph

5.3.8.2 Defines and Types

Table 5-15. USB_Mouse Module Defines and Types

<table>
<thead>
<tr>
<th>Define/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB_MOUSE_DEVICE</td>
<td>This defines the USB Mouse Device ID in the USB Descriptors.</td>
</tr>
<tr>
<td>USB_MOUSE_ENDPOINT</td>
<td>This defines the USB Endpoint in the USB Descriptors.</td>
</tr>
</tbody>
</table>

5.3.8.3 Functions

Table 5-16. USB_Mouse Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usb_init</td>
<td>static</td>
<td>This function initializes the USB User Module and waits for USB to set the device configuration before returning.</td>
</tr>
<tr>
<td>USB_idle</td>
<td>static</td>
<td>This function initializes the USB User Module and waits for USB to set the device configuration before returning.</td>
</tr>
</tbody>
</table>
5.3.9 PDC9198

This module is used to implement platform specific code. Currently the pdc9198.c file is empty. The pdc9198.h file contains the entire platform specific defines for pin and port assignments for a specific feature. Porting from one platform to another should only require modifications to these two files assuming no other features are added or removed.

5.3.9.1 Include Dependency Graph

This graph shows which files directly or indirectly includes PDC9198.h:

Figure 5-15. PDC9198.h Include Graph

5.3.10 PSOCCONFIGTBL

This module is an assembly file and is generated automatically by the Device Editor in PSoC Designer. It is mentioned here because the default values for the port pins are defined here.

5.3.11 ISR

The purpose of this module is to handle GPIO interrupt handling. It provides a single entry point for all GPIO interrupts and calls functions based upon which interrupt is enabled. The enCoRe II chip does not provide the ability to determine which GPIO pin generated the interrupt, but can only determine the port.

5.3.11.1 Include Dependency Graph

This graph shows which files directly or indirectly includes isr.h.

Figure 5-16. isr.h Include Graph
This graph shows which files directly or indirectly includes isr.c.

Figure 5-17.  isr.c Include Dependency Graph

5.3.11.2  Defines and Types

Table 5-17.  ISR Module Defines and Types

<table>
<thead>
<tr>
<th>Define/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM_PORTS</td>
<td>This defines the number of ports that will enable GPIO interrupts.</td>
</tr>
<tr>
<td>MOUSE_ISR</td>
<td>This type is used to identify which isr to enable or disable.</td>
</tr>
<tr>
<td>ISR_PORT0_ENABLE</td>
<td>This macro function sets the interrupt mask bit for Port 0.</td>
</tr>
<tr>
<td>ISR_PORT0_DISABLE</td>
<td>This macro function disables the interrupt mask bit for Port 0.</td>
</tr>
</tbody>
</table>

5.3.11.3  Variable Definitions

Table 5-18.  ISR Module Variable Definitions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled_isrs</td>
<td>The current ISRs that are enabled.</td>
</tr>
</tbody>
</table>

5.3.11.4  Functions

Table 5-19.  ISR Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isr_enable</td>
<td>external</td>
<td>This function enables an interrupt handler at its associated port.</td>
</tr>
<tr>
<td>isr_disable</td>
<td>external</td>
<td>This function disables an interrupt handler at its associated port.</td>
</tr>
</tbody>
</table>

5.3.12  PORT

This module is designed and implemented to solve the problem of doing read-modify-writes on ports that use either pull-up or pull-down resistors on pins configured for input and output. An example of this in the mouse is the Z wheel where pull-ups are used to drive the wheel. The problem occurs when other functions share the same port and need to do a read-modify-write. If the Z wheel happens to be in a state where the value read back is zero, then a zero is written back to the port and disables the pull-ups on the Z wheel. The module caches the drive value for a port and provides an API to consistently maintain the proper drive on a port.
5.3.12.1 Include Dependency Graph

This graph shows which files directly or indirectly includes port.h.

Figure 5-18. port.h Include Graph

This graph shows which files directly or indirectly includes port.c.

Figure 5-19. port.c Include Dependency Graph

5.3.12.2 Defines and Types

Table 5-20. Port Module Defines and Types

<table>
<thead>
<tr>
<th>Define/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT</td>
<td>This type is used in conjunction with the enumerated port values defined in \textit{PLATFORM_H}. The current implementation only supports PORT0.</td>
</tr>
</tbody>
</table>

5.3.12.3 Variable Definitions

Table 5-21. Port Module Variable Definitions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port0_state</td>
<td>The current drive state for port 0.</td>
</tr>
</tbody>
</table>

5.3.12.4 Functions

Table 5-22. Port Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port_init</td>
<td>External</td>
<td>This function initializes the port module.</td>
</tr>
<tr>
<td>port_drive_on</td>
<td>External</td>
<td>This function sets a port bit to a 1 state. The port bits are defined in \textit{PLATFORM_H}: PORT_BIT0, and so on.</td>
</tr>
<tr>
<td>port_drive_off</td>
<td>External</td>
<td>This function sets a port bit to a 0 state. The port bits are defined in \textit{PLATFORM_H}: PORT_BIT0, and so on..</td>
</tr>
</tbody>
</table>
5.3.13 Timer

The timer module provides implementations for the MSTimer ISR and PIT ISR. This module requires
the use of the enCoRe II MSTimer User Module and the PIT User Module. The delay function used
for millisecond timing provides at least the delay requested with no more than one additional milli-
second of delay.

The millisecond delay function puts enCoRe in sleep mode for the duration of the requested delay.
The microprocessor wakes just long enough to update the tick every millisecond and check if the
delay is met and then returns to sleep mode if it has not.

5.3.13.1 Include Dependency Graph

This graph shows which files directly or indirectly includes timer.h

Figure 5-20. timer.h Include Graph

![timer.h Include Graph](image)

This figure shows the include dependency graph for timer.c.

Figure 5-21. timer.c Include Dependency Graph

![timer.c Include Dependency Graph](image)
5.3.13.2 Defines and Types

Table 5-23. Timer Module Defines and Types

<table>
<thead>
<tr>
<th>Define/Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_STAMP</td>
<td>Defines the type of the time stamp.</td>
</tr>
<tr>
<td>USB_ACTIVITY_TIMEOUT</td>
<td>Defines the time to wait for USB Suspend detection; currently set at 3 ms.</td>
</tr>
<tr>
<td>PIT_SLEEP_FACTOR</td>
<td>This is constant definition. The LPO runs a little faster in sleep mode then</td>
</tr>
</tbody>
</table>
<pre><code>                       | awake, so this value is added to PIT reload to tweak it's accuracy.          |
</code></pre>
<p>| SLEEP_IDLE_TICK_COUNT     | Used in USB Remote wakeup. This defines the value for time mouse is completely asleep. |
| SLEEP_OPTICAL_TICK_COUNT  | Used in USB Remote wakeup. This defines the amount of time required for the optical sensor come alive. |
| PIT_MAX_VALUE             | Defines a mask for the max possible PIT value.                              |</p>

5.3.13.3 Variable Definitions

Table 5-24. Timer Module Variable Definitions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sleepTicks</td>
<td>This variable keeps track of the number of PIT ISR has fired.</td>
</tr>
<tr>
<td>calibrateDone</td>
<td>This variable identifies if the timer_calibrate_timer() is called to calibrate the PIT.</td>
</tr>
<tr>
<td>delay_counter</td>
<td>This variable is decremented every millisecond. It is used by timer functions to provide a millisecond counter.</td>
</tr>
<tr>
<td>usb_activity_timeout</td>
<td>This variable is used to identify suspend state on the USB bus. Every millisecond this variable is decremented if no USB activity is detected, otherwise it is reset to USB_ACTIVITY_TIMEOUT.</td>
</tr>
<tr>
<td>usbActivity</td>
<td>This variable is used to hold the return value for the call to USB_bCheckActivity().</td>
</tr>
</tbody>
</table>

5.3.13.4 Functions

Table 5-25. Timer Module Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Linkage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSTIMER_ISR</td>
<td>External</td>
<td>This function is the interrupt service routine for MS Timer.</td>
</tr>
<tr>
<td>PITIMER12_ISR</td>
<td>External</td>
<td>This function is the interrupt service routine for the PIT.</td>
</tr>
<tr>
<td>timer_init</td>
<td>External</td>
<td>This function initializes the tick counter and starts the timer.</td>
</tr>
<tr>
<td>timer_delay_msec</td>
<td>External</td>
<td>This function delays for the number of requested milliseconds plus up to one additional millisecond.</td>
</tr>
</tbody>
</table>

5.4 Verify Output

1. Connect the mouse to the PC and wait for the device to enumerate. When enumeration is done, start using the mouse.
2. Move the mouse in circular, horizontal, and vertical motions.
3. Click the left button to check the left button functionality of mouse.
4. Click the right button to check the right button functionality of mouse.
5. Move the scroll wheel and check the scroll wheel functionality.

Confirm that all the mouse movements are smooth.
A. Appendix

A.1 Schematic

Figure A-1. Mouse Schematic
A.2 Board Layout

Figure A-2. Mouse Bottom
### A.3 Bill of Materials

Table A-1. BOM

<table>
<thead>
<tr>
<th>No</th>
<th>Qty</th>
<th>CY Part Number</th>
<th>Reference</th>
<th>Description</th>
<th>Manufacturer</th>
<th>Mfr Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>710-12016</td>
<td>C1</td>
<td>CAP 10UF 25V ALUM LYTIC RADIAL</td>
<td>Panasonic - ECG</td>
<td>ECA-1EM100</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>750-12013</td>
<td>C2, C4</td>
<td>CAP CERAMIC MONO .1UF 50V 10%</td>
<td>Panasonic - ECG</td>
<td>ECU-S1H104KB</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>740-13176</td>
<td>C3</td>
<td>CAP: 1000 pF, 1KV, RAD, CER</td>
<td>Panasonic</td>
<td>ECK-A3A102KB</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>710-13004</td>
<td>C5</td>
<td>CAP 2.2UF 50V ALUM LYTIC RADIAL</td>
<td>Panasonic - ECG</td>
<td>ECA-1HM2R2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>800-12004</td>
<td>D2, D1</td>
<td>LED RED CLEAR TH</td>
<td>Agilent Technologies</td>
<td>HLMP-ED80-KP000</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>200-12001</td>
<td>EN1</td>
<td>ENCODER CHICONY MOUSE</td>
<td>Chicony</td>
<td>CHI-ENCODER</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>200-12000</td>
<td>J2</td>
<td>HDR mouse 5 pin keyed</td>
<td>Chicony</td>
<td>CHI-HDR5-VERT-KEY</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>800-11997</td>
<td>Q1</td>
<td>IC TRANS PNP SS GP 200MA TO-92</td>
<td>Fairchild Semiconductor</td>
<td>2N3906TFR</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>660-13007</td>
<td>R3</td>
<td>RES 33 OHM 1/8W 5% TH</td>
<td>Yageo</td>
<td>CFR-12JB-33R</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>660-13008</td>
<td>R4</td>
<td>RES 100K OHM 1/8W 5% TH</td>
<td>Yageo</td>
<td>CFR-12JB-100K</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>660-13005</td>
<td>R5</td>
<td>RES 1K OHM 1/8W 5% TH</td>
<td>Yageo</td>
<td>CFR-12JB-1K</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>660-13008</td>
<td>R6</td>
<td>RES 390 OHM 1/8W 5% TH</td>
<td>Yageo</td>
<td>CFR-12JB-390R</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>200-12002</td>
<td>S1, S2, S3</td>
<td>SWITCH 6MM VERT</td>
<td>Chicony</td>
<td>CHI-SW-PB-VERT</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>CY7C63813-PXC</td>
<td>U1</td>
<td>enCoRe II LOW-SPEED USB/PS2 - DIP18</td>
<td>Cypress Semiconductor</td>
<td>CY7C63813-PXC</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>500-11999</td>
<td>U2</td>
<td>AGILENT OPTICAL MOUSE SENSOR DIP8</td>
<td>Agilent Technologies</td>
<td>ADNS-2620</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>NA</td>
<td>W1, W2</td>
<td>WIRE JUMPER TH 400 MIL</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>800-11996</td>
<td>Y1</td>
<td>CER RES 24.000 MHz TH</td>
<td>Murata</td>
<td>CSALS24M0X53-B0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>PDC-9198 REV *A</td>
<td>Reference</td>
<td>Schematic, PCA 121-19800 rev *C for PDC-9198 Rev *A</td>
<td>As per schematic file</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>920-11206</td>
<td>Reference</td>
<td>PCB label, Year, Work-week, SN</td>
<td>Cypress Semiconductor</td>
<td>PCD-9198 Rev *A</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>PDC-9198.dsn</td>
<td>Reference</td>
<td>Schematic, PCA 121-19800 rev *C for PDC-9198 Rev *A</td>
<td>As per schematic file</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>920-11939</td>
<td>Reference</td>
<td>PCB label, PCA revision</td>
<td>121-19800 °C</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>No Load</td>
<td>C6</td>
<td>No Load</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>No Load</td>
<td>R1, R2</td>
<td>No Load</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>