CY4632 Mouse Firmware
User's Guide

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
408-943-2600

Oct 1, 2004

CY4632 Mouse Firmware User's Guide
Table of Contents

Table Listings

Figure Listings

1. INTRODUCTION

1.1 Scope

The audience for this document is intended to be firmware developers that desire to understand and modify the firmware on the Wireless USB LS mouse.

1.2 Overview

This document will cover the design goals, architecture, firmware source code modules and configuration options for the Wireless USB LS mouse. It will not cover the details of the radio subsystem or the configuration options that go with it. Please refer to the CY3632 Wireless USB LS Development kit for theory of operation and details related to the radio subsystem.

1.3 Design Goals

Several design goals drive additional requirements for the mouse firmware. Some of these are architecture related, while others are feature related.

The CY4632 Reference Design Kit utilizes a PSoC controller for the RDK Mouse (Cypress part #: CY8C27443-24PVXI). All references to PSoC in this document refer to the CY8C27443-24PVXI PSoC. Please contact your local sales representative for more information on this PSoC controller.

The architecture was designed to be modular for extendibility and maintainability. It was also designed so that it could easily be ported from one hardware platform to another assuming the use of a PSoC microprocessor. While porting to another microprocessor will require more work, the hardware design was done to minimize use of advanced PSoC features to expedite this effort.

Design efforts have been made to reduce the on time of the microprocessor and radio to conserve battery life. This includes protocol optimizations along with using sleep features of the radio, mouse optics and PSoC microprocessor.

2. DEFINITIONS

Following are some definitions of acronyms and words found in this document. There may be other meanings to these definitions outside of this document.

Bridge – The Bridge is the receiving radio and USB hardware that connects to the PC and enumerates as a Human Interface Device.

Device – The reference to device in this document means the mouse that is sending packets via radio to the bridge.

DVK – A development kit produced by Cypress Semiconductor for showcasing Cypress products with a working development environment.

HID – Is an abbreviation for Human Interface Device and is a product that allows an individual to interface with a computer. A mouse and a keyboard are HID devices.

LED – Is an abbreviation for Light Emitting Diode. In this document it is used in reference to the optical light that detects motion over a textured surface.

PSoC – A configurable integrated circuit comprising of analog and digital blocks and an M8C microprocessor.

RDK – A reference design kit produced by Cypress Semiconductor and used by 3rd parties to produce off-the-shelf products. Everything required to take a product to production is included in the kit. This document is part of the CY4632 Mouse/Keyboard RDK.

USB – The acronym for Universal Serial Bus; a serial standard used in the computing world.

3. HARDWARE OVERVIEW

3.1 Schematic of PDC-9166 Board

The schematic of the PSoC/Radio board (PDC-9166) can be found on the CY4632_RDK release CD at \Hardware\RDK Mouse\LS RDK Mouse Sch.pdf. The schematics for the optical sensor board and button/Z wheel board are not included. Please check the ADNS-2030 datasheet for an example schematic.

3.2 RDK Mouse Photographs

Figure 1: Bind Button and Battery Compartment

Figure 1 shows the location of the bind button. The button is the gray portion in the recessed hole. It can be pressed with a pen tip or similar instrument. There are three screw holes, one at each end (visible in the picture) and one in the battery compartment (hidden under the batteries.) The top of the mouse can be removed once the three screws on the bottom have been extracted.

Figure 2: Exploded Mouse View

Figure 2 is a picture of the mouse with the top removed. All three boards: optical board (PDC-9081), button board (PDC-9080) and radio/PSoC board (PDC-9166) are shown. The PDC-9166 board sits in the center and attaches to the optical board and button board.

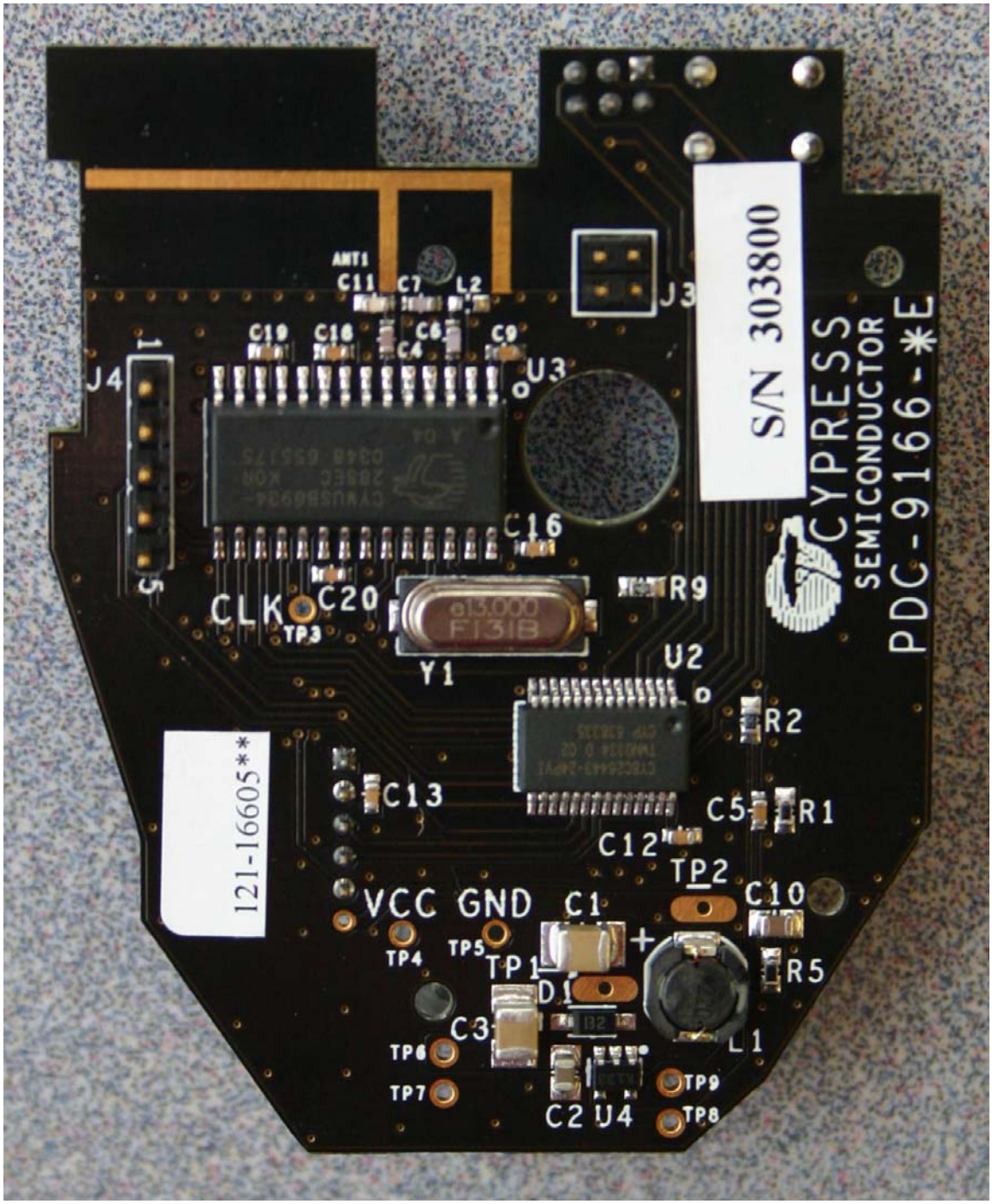

Figure 3: Radio & PSoC Board (PDC-9166)

Figure 3 is a picture of the main controller board with the PSoC and Wireless USB LS Radio. The “F” trace is the antenna. The schematic referenced in section 3.1 is for this board.

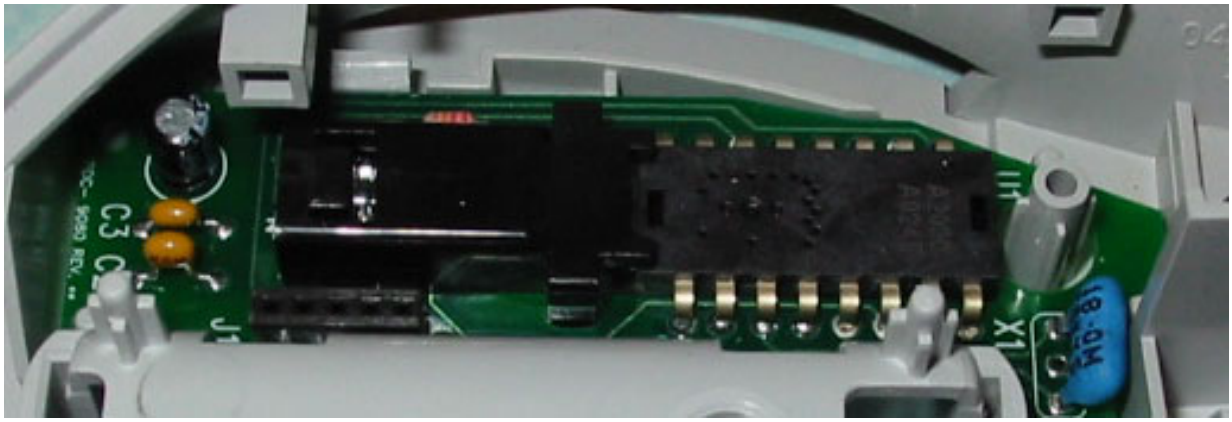

Figure 4: Optical Sensor Board (PDC-9080)

Figure 4 is a picture of the optical sensor board. The Agilent ADNS-2030 Low Power Optical Mouse Sensor, LED housing and connector to the PDC-9166 board are visible in this picture.

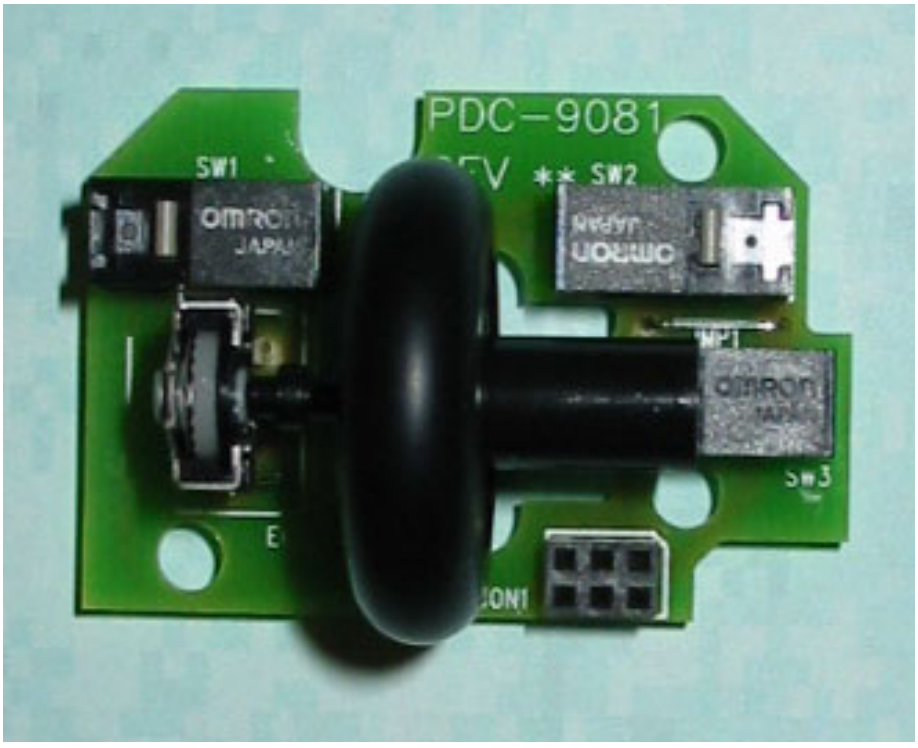

Figure 5: Z Wheel and Button Board (PDC-9081)

Figure 5 is a picture of the button board showing Z Wheel/middle button, left button and right button. The connector to the PDC-9166 board is also visible.

4. DEVELOPMENT ENVIRONMENT

4.1 Tools

The following tools are used to develop the RDK mouse firmware and can be purchased from Cypress Semiconductor.

- PSoC Development Tools Solution
  - PSoC Designer Software version 4.1 SP1
  - PSoC ICE-4000
- ICCM8C C Compiler version 1.28
- Relevant PSoC POD Kit

A Microsoft Windows based PC is used for tool execution.

4.2 PSoC Configuration

The PSoC is configured with one 8-bit timer module as a clock and an SPI module for radio communication. An optional UART module may be configured and used as a debug port. Not all PSoC features are used in order to maintain compatibility with other microprocessor implementations.

5. FIRMWARE ARCHITECTURE

5.1 Model

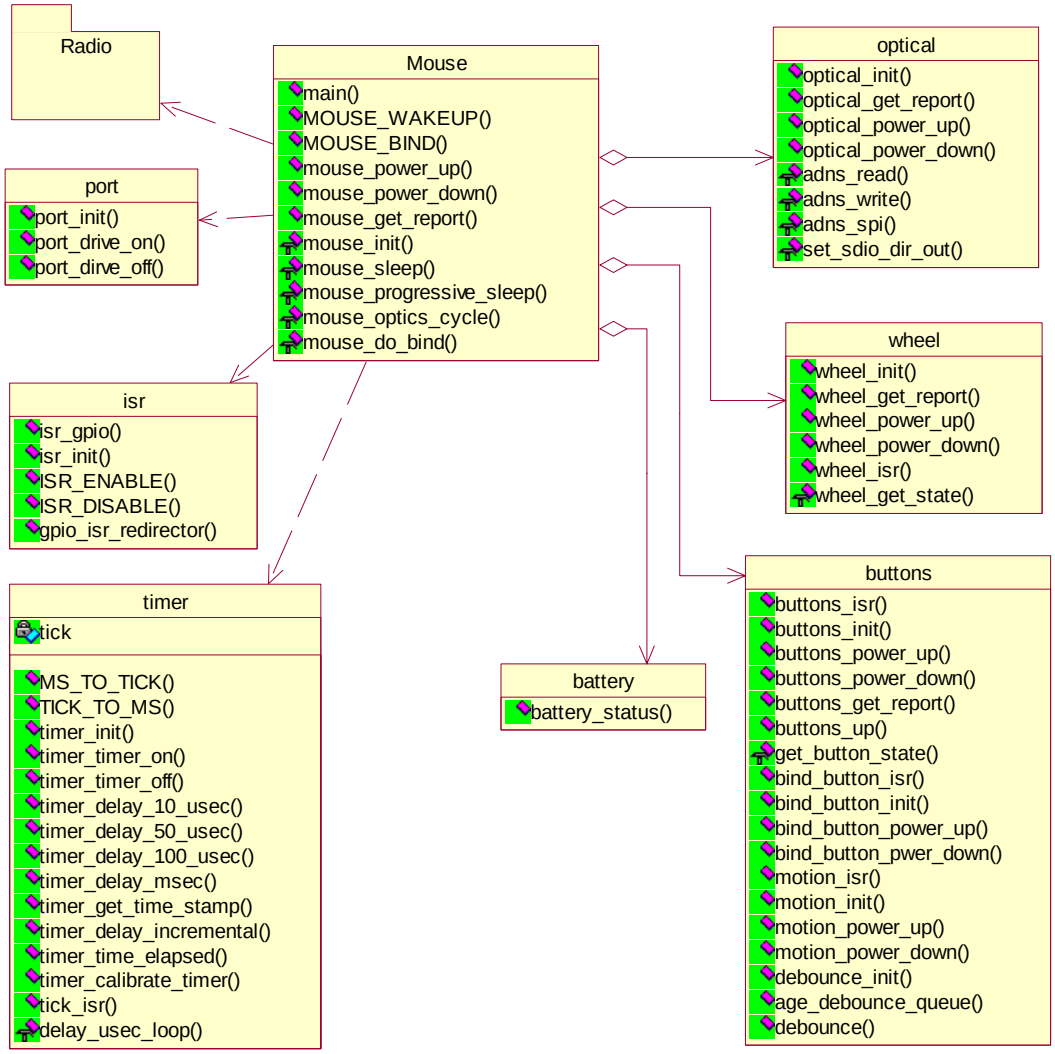

Figure 6: RDK Mouse Architecture

5.2 Normal Mouse Operation

Normal operation of the mouse is initiated by plugging the Wireless USB Bridge into a PC and waiting for it to enumerate. Once this is complete, insert two fresh AA batteries into the mouse. In order for

the mouse to communicate with the bridge, they need to execute a "binding" process. Press the "bind" button on the Wireless USB Bridge and the "bind" button on the mouse, within approximately 5 seconds of each other, to initialize the "binding" process. The mouse will communicate with the Bridge and establish a connection. The mouse is now ready for normal operation.

5.3 Platform & Architecture Portability

The mouse firmware was designed to be easily ported from one hardware platform to another platform with a simple re-mapping of pins on the PSoC. The file *pdc9166.h* maintains the pin mapping definitions that are used throughout the code and is included in about every file by using the macro *PLATFORM_H* that is defined in *appconfig.h*.

Porting the code to another microprocessor architecture requires modification or leverage of the existing code for processor specific features, along with pin definitions.

5.4 Initialization

Initialization of the PSoC chip is done by code that is generated in *boot.asm* by the PSoC Designer software. The module *boot.asm* calls main once the PSoC has been configured and initialized.

Main initializes the components of the mouse along with timer, isr and radio modules. It then goes into a continuous loop monitoring mouse activity and performing a compile-time selectable sleep algorithm.

5.5 Configuration Options

All configuration options for the application can be found in the *appconfig.h* file. There are other configuration options, pertaining to the radio subsystem, which can be found in the *ls_config.h* file. A description of these options can be found in the *CY4632 Protocol Library* document.

5.5.1 MOUSE_POLL_IN_MS

This configuration value sets the period at which the firmware polls the hardware for mouse events to transmit over the radio. This poll period is only active when the mouse has not entered one of the sleep algorithms. The default value is 10 milliseconds. The PSoC

has an internal oscillator that is used for the poll period. Therefore, the poll period is affected by voltage and temperature and may not be exactly the value defined for this configuration value. Setting this value to something smaller than the time it takes to transmit a packet will essentially make the transmit loop repeat as fast as possible.

5.5.2 MOUSE_OPTIC_OFF_TIME_MS

This configuration value sets the time that the mouse must sit idle without detecting any mouse movement before it activates the sleep algorithm. The maximum value that can be defined for this option is 65535 milliseconds.

5.5.3 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 radio. 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.

This configuration value sets the debounce time for buttons that are pressed. It is measured in units of the poll rate. For example, if *MOUSE_DEBOUNCE_COUNT* is set to 3 and *MOUSE_POLL_IN_MS* is set to 10, then the button debounce time will be 30 milliseconds. The default setting is 3.

5.5.4 MOUSE_TX_TIMEOUT

The transmit loop in the mouse attempts to guarantee delivery of mouse events. This loop will eventually time-out if it does not receive a response from the bridge. The time-out time is specified by *MOUSE_TX_TIMEOUT* and is in milliseconds. The default value is 2000 (2 seconds.) Setting this value to 0 will effectively disable the guaranteed delivery feature.

5.5.5 PLATFORM_H

This configuration value identifies the header file that has the platform configuration information. The default value is *pdc9166.h*, which is the identifier for the mouse board that is shipped with the

RDK. It is anticipated that this macro will change when the code is ported to another platform.

5.5.6 MOUSE_MOTION_NOT_TIME_SLEEP

This configuration definition is used to selectively compile code for one of two mouse sleep algorithms. If this value is defined, then the sleep algorithm that uses the motion sensor is used. If it is not defined then a time-based algorithm that modifies the duty cycle of the optical LED is used. See section 5.6 for further detail.

5.5.7 MOUSE_800_NOT_400_CPI

This configuration definition is used to select between 800 or 400 counts per inch when configuring the optical chip. If it is defined then 800 cpi is selected. If it not defined then 400 cpi is selected.

5.5.8 MOUSE_POWER_ON_BIND

Enabling this option will cause the mouse to enter bind mode when batteries are inserted. It will continue to enter bind mode when batteries are inserted until it has successfully bound with a Bridge and saved the bind parameters in Flash. Once the mouse has been bound to a bridge, this feature will be disabled.

5.5.9 TIMER_CAL

This configuration definition is used to selectively compile in the one-millisecond timer calibration support. The timer is calibrated at power-on and when the mouse exits sleep mode.

5.5.10 MOUSE_BATTERY_STATUS

Enabling this feature will cause the battery level measurement code to be compiled into the mouse image. The mouse will then measure battery level at power-on and before entering sleep mode and will then report it when requested by the Bridge.

5.5.11 MFG_TEST_CODE

This configuration definition is used to selectively compile in the manufacturing test code.

5.5.12 MFG_ENTER_BY_KEY_NOT_PIN

This configuration definition is used to select whether the manufacturing test code is executed by pulling a PSoC pin to ground or by the test mode module. When this value is defined, then the manufacturing test code may be executed by entering test mode 1. If it is not defined, then the manufacturing test code may be executed by pulling a defined port pin to ground within 100 milliseconds of power on. See section 6.7.1 for the port pin definition.

5.5.13 MOUSE_TEST_MODES

This configuration definition is used to selectively compile code for mouse test modes. If this value is defined, then test modes are compiled into the executable image. If it is not defined, then the test mode code is omitted.

5.5.14 MOUSE_TEST_MODE_1

This definition will selectively compile code for test mode 1. Test mode one, when entered, will activate the manufacturing test mode. The only way to exit this test mode is by cycling power. This test mode is disabled if *MOUSE_TEST_MODES* is not defined.

5.5.15 MOUSE_TEST_MODE_2

This definition will selectively compile code for test mode 2. This test mode will repeatedly move the mouse in a square box without any button presses. This test mode can be used anywhere on the PC screen since it only does mouse movement. This test mode is disabled if *MOUSE_TEST_MODES* is not defined.

5.5.16 MOUSE_TEST_MODE_3

This definition will selectively compile code for test mode 3. This test mode will move the mouse in a fashion to repeatedly draw the letters “USB” in a drawing program. Mouse acceleration or advanced motion should be turned off when performing this test. This test mode is disabled if *MOUSE_TEST_MODES* is not defined.

5.6 Sleep Algorithms

Two algorithms have been implemented in this RDK. The selection of which is done by a compile time option explained in section 5.5.6

The first algorithm makes use of a motion detection circuit. When the firmware detects that the mouse has been idle for a configurable amount of time, as explained in section 5.5.2, it puts the PSoC and optics into sleep mode. When the PSoC is asleep, an interrupt is required to wake it up and start execution of the code again. A button press or the detection of motion from the motion circuit is the only two sources that provide this interrupt. The timer module is also turned off during sleep mode to help conserve battery power so notion of time is lost. This sleep algorithm is always compiled into the code since it is a subset of the other option that is explained next.

The “time” based algorithm is one where the PSoC remains active and power is conserved by changing the on-time or duty cycle of the optical LED after the idle time as explained in section 5.5.2 has elapsed. The duty cycle gets smaller as time passes and changes in three steps. Once the final step is met, the PSoC and optics are put into sleep mode where only a button press will wake the firmware. This algorithm is provided for when it is not desired to provide a motion detection circuit.

See section 6.1 for the implementation of these algorithms.

5.7 Radio Subsystem

The radio subsystem is composed of the following modules and provides an API to the application for sending data over the radio.

- bind
- protocol
- radio
- spi
- flash

Please refer to the *CY4632 Protocol Library* documentation for theory of operation along with a detailed description of the API and configuration options. The *spi* module is coupled to the platform in order to communicate with the radio.

5.8 Mouse Wireless Protocol

The mouse protocol has been optimized to reduce the “on-time” of the radio, which equates to reduced power consumption. This optimization relies upon the RDK requirement of a three-button mouse. With this requirement, it is possible to combine the Z axis and the button report into a single byte, allowing five bits of information for the Z axis and three bits for the buttons.

The radio library offers the ability to send variable length packets, allowing the opportunity to reduce the number of bytes transmitted over the air, in order to extend battery life.

Since mouse usage data demonstrates that X, Y axis data is more frequent than Z axis or button presses, the following transmission packet formats are implemented in this RDK. The packet formats only show the application payload and do not show the radio protocol overhead of a packet header and CRC byte.

5.8.1 Packet Format 1

When there is only X, Y delta data, then the transmitted packet will be two bytes.

Table 1: Packet Format 1

Byte 1	Byte 2
X Delta (8 bits)	Y Delta (8 bits)

5.8.2 Packet Format 2

When there is either Z delta data or a button is pressed, then the transmitted packet will be three bytes. In the case where there is no X, Y delta data, but there is Z delta or button data, then the X, Y delta bytes will be set to zero. The Z delta is a signed value with bit 4 as the sign bit. When a button is held down, this packet format will continue to be sent at the poll rate defined in section 5.5.1 until the button is released. This is so the bridge can generate a button up event if the connection is lost.

Table 2: Packet Format 2

Byte 1	Byte 2	Byte 3
X Delta	Y Delta	Buttons (bits 765),

(8 bits)	(8 bits)	Z Delta (bits 43210)

5.8.3 Packet Format 3

When battery voltage level is communicated, then the transmitted packet will be 1 byte.

Table 3: Packet Format 3

Byte 1
Battery Level (1 – 10)

5.9 Test Modes

This RDK provides a compile-time option of adding test modes to the mouse; see section 5.5.7 for enabling this option. The test mode module is implemented in a way that it can be easily extended to add other test modes. Currently there are several test modes supported in the *testmode* module. When this option is not enabled then all test mode code is removed from the compilation.

The test selection is initiated by holding down the left and right buttons at the same time while inserting the batteries. The optical LED will come on for a short time and then turn off to conserve battery power. Once the LED has turned off the two buttons may be released. Then, pressing the left button once, will select test mode one, twice will select test mode two, and so on. Press the right button to activate the test.

Test mode one performs a manufacturing test. The only way to exit this test mode is to cycle power.

Test mode two continuously moves the mouse pointer in the pattern of a square box. The “pen” is not down so this test may be performed over most non-drawing windows. However, the right mouse button is used for indexing so a pop-up window may periodically appear. Once the test mode has been entered, pressing the left button will initiate the drawing sequence and turn off the optical LED. Pressing the left button again will stop the drawing sequence. The mouse can then be moved to a different location and drawing started again by pressing the left button. The

test mode is exited by pressing the right mouse button when the mouse is not in the drawing sequence. The mouse will then resume with normal operation.

Test mode three continuously performs a vector drawing test within a drawing application. Once the mode has been entered as described above, then the optical LED comes back on so that the mouse can be positioned over the drawing window. Pressing the left button will initiate the drawing sequence and turn off the optical LED. Pressing the left button again will stop the drawing sequence. The mouse can then be moved to a different location and drawing started again by pressing the left button. The test mode is exited by pressing the right mouse button when the mouse is not in the drawing sequence. The mouse will then resume with normal operation.

The output of this test mode is the letters “WirelessUSB” being drawn and redrawn on top of itself continuously. The mouse “acceleration” or “enhance pointer precision” option needs to be turned off for this test to execute properly. If the letters are drawn erratically with uneven sides or excessive amounts of space in between them, then check this setting or it’s equivalent (based upon your PC operating system.) The size of the letters can be adjusted by setting the mouse “speed”.

5.10 Flash Security

The mouse project within PSoC Designer has a file called *FlashSecurity.txt*. This file specifies access rules to blocks of the Flash ROM. Please see the documentation at the top of the file for definitions. This file is shipped with its default configuration in the RDK with the exception of one change. The block starting at address 3E80 hex has been changed from W: Full (Write protected) to U: Unprotected. This location of Flash has been dedicated to saving non-volatile configuration values for the radio subsystem.

5.11 Battery Monitor

The battery monitor circuit is implemented using an RC circuit connected to two PSoC pins. The firmware has been tuned to provide a ten level measurement of the battery voltage. The battery level is measured just before the mouse enters sleep mode. The value is then saved and transmitted to the bridge upon request. The battery level is also measured when batteries are inserted into the mouse.

The process starts by first setting the PSoC pins *BATT_LEV1* and *BATT_LEV2* to high impedance inputs and allowing the RC circuit to reach a steady state. This allows *BATT_LEV1* to reach the same potential as the battery with the capacitor. *BATT_LEV2* is then driven to ground and the RC time constant is measured to the point at which the *BATT_LEV1* detects a logic 0 state. This time measurement is the battery voltage measurement. Next, both *BATT_LEV1* and *BATT_LEV2* are driven to a logic 1 state initializing the RC circuit for a VCC measurement. *BATT_LEV1* is then set to a high impedance input and *BATT_LEV2* is driven to ground. Once again, the RC time constant is measured to the point at which the *BATT_LEV1* detects a logic 0 state. This time becomes a VCC reference measurement.

The firmware then uses the battery measurement and VCC measurement to compute a battery level between 1 and 10 inclusive.

6. CODE MODULES

Following are descriptions of the module contents. For concepts of operation, see section 5.

6.1 MOUSE

The mouse module contains the main entry point, a routine for data acquisition from the various components of the mouse, sleep algorithms, and management of report frequency to the Bridge.

6.1.1 Defines & Types

Table 4: Mouse Module Defines & Types

Define/Type	Description
TX_PACKET	This structure defines the data payload portion of the radio transmit packet. In order to support variable length packets, the *optical* and *battery_level* are a union. The *combi* is used to combine the ZWheel and button presses into one byte.
MOUSE_SLEEP	This conditional compile macro selects the sleep algorithm to be used on the mouse.
MOUSE_WAIT_BUTTON_UP	This conditional 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*.
MOUSE_SLEEP	This macro defines the sleep function to be used when exiting active mode.
MOUSE_PROTOCOL_BIND	This macro is used to call the bind function for two-way communication.
DEBUG_INIT	This macro initializes the debug module. Care should be used when turning this function on as it modifies mouse timing and performance.
CACHE_BATTERY_STATUS	This macro is used to save battery level when the battery status feature is enabled.

6.1.2 Variable Definitions

Table 5: Mouse Module Variable Definitions

Variable	Description
sleep_timer	This variable is a time stamp of the last mouse activity. It is used to determine when to enter sleep mode.
mouse_asleep	This flag is changed in interrupt context to wake the mouse from sleep mode.
mouse_battery_status	This is used capture and save the battery level until the next transmission.
mouse_auto_bind	This is a flag set in interrupt context and is used to request that the mouse perform the bind process.

6.1.3 Functions

Table 6: Mouse Module Functions

Function	Linkage	Description
main	external	This function is the main entry point for the application. It initializes all mouse application components and radio components and executes the poll loop. Radio

		transmission takes place when there are mouse events to send at each poll period.
MOUSE_WAKEUP	external	This macro function is called from interrupt context and requests the mouse to exit sleep mode.
MOUSE_BIND	external	This macro function is called from interrupt context and requests the mouse to perform a bind process.
mouse_power_up	external	This function performs the power on sequence for the mouse when resuming from sleep mode.
mouse_power_down	external	This function performs the power down sequence for the mouse before entering sleep mode.
mouse_get_report	external	This function polls the optics, wheel and buttons for potential events to send over the radio. It also builds the packet payload.
mouse_init	static	This function initializes all components of the mouse application.
mouse_sleep	static	This function performs a deep sleep of all mouse hardware. It is used in both sleep algorithms.
mouse_progressive_sleep	static	This function implements the time based sleep algorithm.
mouse_optics_cycle	static	This is a helper function for the time base sleep algorithm. It handles the duty cycle on the optical LED.
mouse_do_bind	static	This function performs the bind process by calling upon the radio subsystem services.

6.2 OPTICAL

The optical module provides the low level routines for communicating with the optical sensor chip. Standard PSoC 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.

6.2.1 Defines & Types

Table 7: Optical Module Defines & Types

Define/Type	Description
OPTICAL_REPORT	This structure defines the layout of the X and Y delta data within the transmission packet payload.
ADNS_SPI_DIRECTION	This enumeration is used to specify the direction of communication to the optical chip.
Optical Definitions	These definitions are used to interface with the Agilent optical sensor. They define register addresses and bit masks for those registers.

6.2.2 Functions

Table 8: Optical Module Functions

Function	Linkage	Description
optical_init	external	This function resets, establishes communication and initializes the optical sensor.
optical_get_report	external	This function polls the optical sensor for X/Y motion detail and prepares the report. It also handles overflow conditions.
optical_power_up	external	This function brings the optical sensor out of a sleep mode.
optical_power_down	external	This function puts the optical sensor into a sleep mode.
adns_read	static	This function reads an optical sensor register at a given address.
adns_write	static	This function writes a value to a given address in the optical sensor.
adns_spi	static	This function handles the bit shifting of data to and from the optical sensor.
set_sdio_dir_out	static	This is a helper function for the **adns_spi** function and changes port drive characteristics.

6.3 BUTTONS

The buttons module actually contains four components: left, middle and right button code, bind button code, motion detection code and debounce code. This module handles interrupts, debounce and enabling of these various components.

6.3.1 Defines & Types

Table 9: Buttons Module Defines & Types

Define/Type	Description
LEFT_BUTTON_REPORT_BIT	Bit position for the left button in the button report.
RIGHT_BUTTON_REPORT_BIT	Bit position for the right button in the button report.
MIDDLE_BUTTON_REPORT_BIT	Bit position for the middle button in the button report.
NUMBER_OF_BUTTONS	Defines the number of buttons on the mouse.
BUTTON_REPORT	Defines the button report layout for the transmission data packet.
MOUSE_BUTTON_ALL	A simplifying macro used in initialization of all buttons.

6.3.2 Variable Definitions

Table 10: Buttons Module Variable Definitions

Variable	Description
last_button_state	This variable maintains the state of the buttons when last polled.
debounce_queue	This array of values is used for debouncing button presses.

6.3.3 Functions

Table 11: Buttons Module Functions

Function	Linkage	Description
buttons_isr	external	This function is an interrupt handler for when a button is pressed. It is only activated when

		the mouse is asleep and is used to wake-up the mouse when a button is pressed.
buttons_init	external	This function initializes the port, state and interrupt handler for the buttons component.
buttons_power_up	external	This function performs a power up action for the buttons component when coming out of sleep.
buttons_power_down	external	This function performs a power down action for the buttons component before going to sleep.
buttons_get_report	external	This function polls the button status and performs debouncing on any keys that change state. It also formats the report for transmission.
buttons_up	external	This helper function returns true if all buttons are up and false if a button is still held down.
get_button_state	static	This helper function reads the state of the buttons and translates button state to report state.
bind_button_isr	external	This interrupt handler is used to wake-up the mouse, if asleep, and request that the bind process be performed.
bind_button_init	external	This function initializes the port and interrupt handler for the bind button.
bind_button_power_up	external	This function performs a power up action for the bind button after the mouse has been asleep.
bind_button_power_down	external	This function performs a power down action for the bind button before the mouse enters sleep mode.
motion_isr	external	This function is the interrupt handler for detection of change in the motion circuit. It is only enabled when the mouse is asleep

		and will wake-up the mouse when triggered.
motion_init	external	This function initializes the motion detection port and interrupt handler.
motion_power_up	external	This function performs a power up action for the motion detection circuit when the mouse wakes-up.
motion_power_down	external	This function performs a power down action for the motion detection circuit before the mouse goes into sleep mode.
debounce_init	external	This function initializes the debounce logic for the buttons.
age_debounce_queue	static	This function decrements debounce values for all keys every time the mouse buttons are polled. This provides a debounce period for when a button is pressed
debounce	static	This function starts the debounce clock when a button is pressed and blocks out further button changes for the pressed button until the debounce time has expired.

6.4 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.

6.4.1 Defines & Types

Table 12: Wheel Module Defines & Types

Define/Type	Description
WHEEL_REPORT	This structure defines the report for the transmission data payload for the Z wheel.
MOUSE_WHEEL_STATE0	Bit translation for Z wheel port input.
MOUSE_WHEEL_STATE1	Bit translation for Z wheel port input.

WHEEL_STATE	This structure defines Z wheel state data that needs to be persistent.

6.4.2 Variable Definitions

Table 13: Wheel Module Variable Definitions

Variable	Description
wheel_state	This is a state variable used to maintain the state of the Z wheel from interrupt to interrupt until the Z delta is polled.

6.4.3 Functions

Table 14: Wheel Module Functions

Function	Linkage	Description
wheel_init	external	This function initializes the Z wheel port, state data and interrupt handler.
wheel_get_report	external	This function formats the Z wheel delta data for the report and clears state data.
wheel_power_up	external	This function performs a power up action for the Z wheel component when the mouse exits sleep mode.
wheel_power_down	external	This function performs a power down action for the Z wheel before the mouse enters sleep mode. It is required to turn of the drive for the Z wheel when the mouse sleeps to save battery life. The sensor could be left in a power drain position.
wheel_isr	external	This function is the interrupt handler for when transitions take place on the Z wheel pins. It captures the change and records it in the state variable.
wheel_get_state	static	This is a helper function used to read the Z wheel port.

6.5 BATTERY

This module measures the battery voltage level and computes a level from 1 to 10. Time for the RC circuit to stabilize with the battery voltage needs to be given before calling this module.

6.5.1 Functions

Table 15: Battery Module Functions

Function	Linkage	Description
battery_status	external	This function reads a time constant related to the battery voltage and VCC (for reference) and computes a battery level from 1 to 10. See section 5.11 for details of operation.

6.6 TESTMODE

This module provides the implementation for various test modes. The code is easily extended for adding other test modes.

6.6.1 Defines & Types

Table 16: Testmode Module Defines & Types

Define/Type	Description
TEST_EXECUTE	This conditional compile macro makes a call to the test mode code.
TEST_MODES	This type enumerates the test modes implemented in this module.
BUTTON_POLL_MS	This macro defines the poll rate for buttons as part of the user interface to selecting the test mode.
VECTOR_POLL_MS	This macro defines the report rate for the vector draw test modes.

6.6.2 Variable Definitions

Table 17: Testmode Module Variable Definitions

Variable	Description
square_vector_table	This table defines relative mouse movements for the square box draw test mode.
usb_vector_table	This table defines the relative mouse movements for the “WirelessUSB” draw test mode.

6.6.3 Functions

Table 18: Testmode Module Functions

Function	Linkage	Description
test_execute	external	This function is the entry point for selection of a test mode and execution of that test mode. If the left and right buttons are not pressed when this function is called, then the mouse returns to normal operation.
test_get_testmode	static	This function reads the number of left button presses and returns the

		test mode selected.
test_wait_buttons_up	static	This helper function waits for all buttons to be released.
test_power_up	static	This helper function powers the mouse up for use in test modes.
test_power_down	static	This helper function powers portions of the mouse down for use in test modes.
test_mode_1	static	This function implements test mode 1. This test mode invokes the manufacturing test code. There is no way to exit this mode except by a power cycle.
test_mode_2	static	This function implements test mode 2. This test mode moves the mouse pointer in the form of a square without the pen down.
test_mode_3	static	This function implements test mode 3. This test mode does a vector draw of the letters "WirelessUSB" in a drawing window.
vector_draw	static	This is the vector draw function for test mode two and three.

6.7 MFGTEST

This module may be conditionally compiled in to provide manufacturing test support. The module configures the radio for reception and then enters a loop waiting for packets to be sent from the tester. It computes the total number of bits received and the number of invalid bits received. It then sends packets to the tester followed by the previously computed results. After which it re-enters the reception loop waiting for another test cycle. The manufacturing test code may be entered by grounding a microprocessor port pin or by the test mode mechanism defined in section 5.9. The method of entry is a compile time option described in section 5.5.12. The manufacturing test code can only be exited by cycling power.

6.7.1 Defines & Types

Table 19: MFGTest Module Defines & Types

Define/Type	Description
MFG_PN_CODE	This macro defines the PN code used for the manufacturing test. See radio documentation.
MFG_CHANNEL	This macro defines the channel used for the manufacturing test. See radio documentation.
MFG_THRESHOLD_L	This macro defines the threshold when correlating a '0' data bit. See the radio documentation.
MFG_THRESHOLD_H	This macro defines the threshold when correlating a '1' data bit. See the radio documentation.
MFG_PA_BIAS	This macro defines the power amplifier bias used for the test. See radio documentation.
MFG_EOF_TIMEOUT	This macro defines the number of bits that need to be seen before end-of-frame is detected. See radio documentation.
MFG_RX_TIMEOUT	The number of milliseconds between two packets, once packet sending has started, before the receiving loop will timeout.
MFG_NUM_PACKETS	The number of packet expected to receive and send.
MFG_PACKET_PAYLOAD_SIZE	The data size of the packets being received and sent. Does not include header and checksum overhead.
MFG_PRE_TX_DELAY	The time in milliseconds to wait after receiving and before sending out data packets.
MFG_INTER_TX_DELAY	The time in milliseconds to wait in between data packets being sent out.
MFG_NUM_RESULTS_PACKETS	The number of times to send each result packet.
MFG_PRE_RESULTS_DELAY	The time in milliseconds to wait before sending out the results packets.
MFG_INTER_RESULTS_DELAY	The time in milliseconds in

	between each repetition of result packet sent.
MFG_TEST_PORT	The port owning the pin used to enter manufacturing test mode.
MFG_TEST_PIN	The pin used to enter manufacturing test mode. This is used in conjunction with MFG_TEST_PORT.
MFG_TEST_DM0	Drive mode register 0 for the port used in MFG_TEST_PORT. This is used to change drive characteristics during power-up.
MFG_TEST_DM1	Drive mode register 1 for the port used in MFG_TEST_PORT. This is used to change drive characteristics during power-up.
MFG_HDR_TEST	Header byte for test data packets sent to the tester from the device.
MFG_HDR_RES1	Header byte for results packet number one.
MFG_HDR_RES2	Header byte for results packet number two.
MFG_HDR_RES3	Header byte for results packet number three.
MFG_TEST_PKT_SIZE	Total packet size defined as MFG_PACKET_PAYLOAD_SIZE + header byte + checksum byte.
MFG_RES_PKT_SIZE	Packet size for results packets. This is hard coded to 10 bytes and includes header byte and checksum byte.

6.7.2 Functions

Table 20: MFGTest Module Functions

Function	Linkage	Description
mfg_pin_check	external	This function checks the state of the defined manufacturing port pin. If the port pin is low, then the manufacturing test is executed.

mfg_test	external	This is the top-level function for the manufacturing test.
mfg_setup_radio	static	This function initializes the radio to a manufacturing test state.
mfg_receive_test_packets	static	This function receives test packets sent from the tester to the device.
mfg_send_test_packets	static	This function sends test packets from the device to the tester.
mfg_send_results	static	This function sends statistics of data that was received by the device.
mfg_send_result_packet	static	This function performs the actual sending of the results packet with appropriate delays.

6.8 PDC9166

This module is used to implement platform specific code. Currently the *pdc9166.c* file is empty. The *pdc9166.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.

6.9 PSOCGPIOINT

This module is an assembly file and is generated automatically by the PSoC Designer in the lib directory. It is mentioned here because it has been modified after PSoC Designer generated the code. PSoC Designer generates code such that when a GPIO interrupt occurs the vector jumps to this module. This module in turn needs to jump to the *isr_gpio* interrupt service routine provided in the ISR module, see section 6.10. If GPIO interrupts stop working, then this is a good place to check since this is a generated file. See section 6.10 for a description of the function *gpio_isr_redirector*().

6.10 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 PSoC does

not provide the ability to determine which GPIO generated the interrupt.

6.10.1 Functions

Table 21: ISR Module Functions

Function	Linkage	Description
isr_gpio	external	This is the main GPIO interrupt service routine. It calls interrupt handlers if the interrupt is enabled using the *gpio_isr_redirector*() function. The interrupt handlers need to be callable even if they are not the source of the interrupt.
isr_init	external	This function initializes the interrupt enable registers and turns on interrupts for the microprocessor.
ISR_ENABLE	external	This macro function enables an interrupt handler at its associated port pin.
ISR_DISABLE	external	This macro function disables an interrupt handler at its associated port pin.
gpio_isr_redirector	external	This function is used as a mechanism to bypass the compiler's inefficient ISR mechanism of pushing all registers when an ISR makes another function call

6.11 PORT

This module was 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 get 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.

6.11.1 Defines & Types

Table 22: Port Module Defines & Types

Define/Type	Description
PORT	This type is used in conjunction with the enumerated port values defined in *pdc9166.h*: PORT0 through PORT5. However, the current implementation only supports PORT0.

6.11.2 Variable Definitions

Table 23: Port Module Variable Definitions

Variable	Description
port0_state	The current drive state for port 0.

6.11.3 Functions

Table 24: Port Module Functions

Function	Linkage	Description
port_init	external	This function initializes the port module.
port_drive_on	external	This function sets a port bit to a 1 state. The port bits are defined in *pdc9166.h*: PORT_BIT0, etc..
port_drive_off	external	This function sets a port bit to a 0 state. The port bits are defined in *pdc9166.h*: PORT_BIT0, etc..

6.12 TIMER

The timer module provides a one-millisecond tick for the system. The tick resolution can be changed, but is set for one millisecond for the mouse. This module requires the use of a timer block on the PSoC. The delay function used for millisecond timing will provide at least the delay requested with no more than one additional millisecond of delay. The microsecond delay functions have been tuned as best as possible for a 12 MHz clock setting for the microprocessor. The millisecond delay function will sleep the PSoC for the duration of the requested delay. The microprocessor wakes just long enough to update the tick every millisecond and check if the delay has been met and then returns to sleep mode if it has not.

See documentation in the module for requirements on configuring the PSoC block.

6.12.1 Defines & Types

Table 25: Timer Module Defines & Types

Define/Type	Description
TIME_STAMP	This defines the type of the time stamp.
MS_TO_TICK	This macro is used to convert milliseconds to ticks.
TICK_TO_MS	This macro is used to convert ticks to milliseconds.

6.12.2 Variable Definitions

Table 26: Timer Module Variable Definitions

Variable	Description
tick	This variable keeps track of the number of ticks since the timer was turned on.

6.12.3 Functions

Table 27: Timer Module Functions

Function	Linkage	Description
tick_isr	external	This function is the interrupt service routine for the timer expired interrupt. It increments the tick counter.
timer_init	external	This function initializes the tick counter and starts the timer.
timer_timer_on	external	This function turns the timer and system tick on.
timer_timer_off	external	This function turns the timer and system tick off.
timer_delay_10_usec	external	This function delays for 10 microseconds.
timer_delay_50_usec	external	This function delays for 50 microseconds.
timer_delay_100_usec	external	This function delays for 100 microseconds.
delay_usec_loop	internal	This is a helper function used by the delay_xx_usec functions.
timer_delay_msec	external	This function delays for the number

		of requested milliseconds plus up to one additional millisecond.
timer_get_time_stamp	external	This function returns the current tick counter value.
timer_delay_incremental	external	This function takes a time stamp and delays for the incremental time between the time stamp and the requested period. This function is used for meeting poll rate requirements.
timer_time_elapsed	external	This function returns true if the specified amount of time has elapsed since a given time stamp.
timer_calibrate_timer	external	This function calibrates the one-millisecond timer using system clock as a reference.

6.13 TICKINT

This module is an assembly file and is generated automatically by the PSoC Designer in the lib directory. It is mentioned here because it has been modified after PSoC Designer generated the code. PSoC Designer generates code such that when a Timer interrupt occurs the vector jumps to this module. This module in turn needs to jump to the *tick_isr* interrupt service routine discussed in section 6.11.

7. REFERENCES

CY3632 Wireless USB Development Kit documents

CY4632 Protocol Library

PSoC Designer version 4.1 documentation

CY4632 RDK Kit schematics

Agilent ADNS-2030 Low Power Optical Mouse Sensor Data Sheet