



# CY4631 Wired Mouse Reference Design Kit Guide

Spec. # 001-47345 Rev. \*C

Cypress Semiconductor  
198 Champion Court  
San Jose, CA 95134-1709  
Phone (USA): 800.858.1810  
Phone (Intl): 408.943.2600  
<http://www.cypress.com>

## Copyrights

© Cypress Semiconductor Corporation, 2008-2009. 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 Designer™, and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark 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 Data Sheets. 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



<b>1. Introduction</b>	<b>5</b>
1.1 Included in This Kit .....	5
1.2 Overview .....	5
1.2.1 Description .....	5
1.2.2 Block Diagram .....	6
1.2.3 Theory of Operation .....	6
1.3 Document Revision History .....	7
1.4 Documentation Conventions .....	7
<b>2. Hardware</b>	<b>9</b>
2.1 Mouse Overview .....	9
2.2 Major Components .....	9
2.3 Additional Electrical Feature .....	9
2.4 Mechanical .....	10
2.5 HW Documentation .....	10
2.6 Manufacturing Considerations .....	10
<b>3. Firmware</b>	<b>11</b>
3.1 Introduction .....	11
3.2 Development Environment .....	11
3.3 Design Goals .....	11
3.3.1 Mouse Application .....	11
3.3.2 USB Functionality .....	11
3.3.3 Bootloader Functionality .....	12
3.4 Architecture .....	12
3.4.1 User Modules .....	12
3.4.1.1 Full-Speed USB Bootloader User Module .....	12
3.4.1.2 EEPROM User Module .....	16
3.4.1.3 NAV User Module .....	16
3.4.1.4 Timer16 User Module .....	17
3.4.2 Application Code .....	17
3.4.2.1 Global Data Structures .....	17
3.4.2.2 Source Code Files .....	19
3.4.2.3 Source Code Functions .....	20
3.5 Example Project Build .....	23
3.5.1 Project Change .....	23
3.5.2 Step-By-Step Changes .....	23
3.5.2.1 Step 1 - Launch PSoC Designer .....	23
3.5.2.2 Step 2 - Generate the Application .....	24
3.5.2.3 Step 3 - Check for Errors .....	24
3.5.2.4 Step 4 - Open the Appropriate Source File .....	25
3.5.2.5 Step 5 - Edit the Source File .....	25

3.5.2.6	Step 6 - Build the Solution.....	26
3.5.2.7	Step 7 - Launch Bootloader Tools.....	26
3.5.2.8	Step 8 - Rebuild Project with Valid Checksum .....	27
3.5.2.9	Step 9 - Prepare to Program the Device .....	28
3.5.2.10	Step 10 - Program the Device .....	29
3.5.2.11	Step 11 - Test.....	30
<b>4.</b>	<b>Regulatory Tests</b>	<b>31</b>
4.1	Introduction.....	31
4.2	USB .....	31
4.3	FCC .....	31
4.4	ESD .....	31
4.5	WLK.....	31

# 1. Introduction



## 1.1 Included in This Kit

- Full-speed gaming mouse based on the CYONS2100-LBXC Laser Navigation System on Chip
- Cypress MiniProg programmer with USB cable
- PDC-9241 Adapter for programming
- Kit CD, including
  - Hardware Design Files - Schematic, Bill of materials, PCB layout files
  - Firmware - Full source and object code
  - Documentation - Kit Guide, Release Notes, Errata and so on
  - Development Tools - PSoC Designer 5.0™ Service Pack 4.5 and PSoC Programmer

## 1.2 Overview

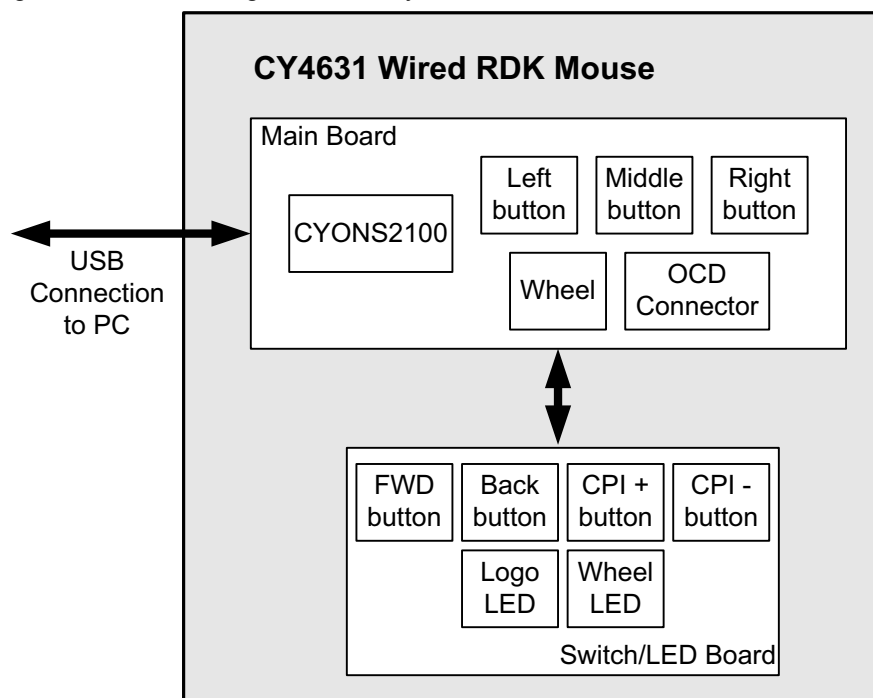
### 1.2.1 Description

The OvationONS II Laser Navigation System on Chip is Cypress Semiconductor's second generation laser product. It combines the powerful Harvard architecture MCU found in Cypress' PSoC devices, with a high-speed, high-precision OptiCheck™ sensor. Full-speed USB capabilities are also included in the package.

The CY4631 OvationONS II wired mouse RDK provides a turnkey reference design suitable for PC mouse manufacturers. The kit incorporates the CYONS2100 OvationONS II Laser Navigation System on Chip in a wired mouse design, complete with PCB and mouse skins. This mouse is a multi-function device, featuring 7 buttons, one single-color logo LED, and one tri-color indicator LED. The system includes a total of 2 printed circuit boards: a main board and a daughter card. The main board contains the OvationONS II SoC, left, right, and center (scroll wheel) buttons along with the USB interface. The daughter card includes the front/back buttons, the DPI adjust buttons, the logo LED, and the tri-color LED. In addition, a MiniProg and USB adapter are included in the kit for programming.

## 1.2.2 Block Diagram

Figure 1-1. Block Diagram of the System



The description of the block diagram in [Figure 1-1](#) is quite simple. A total of two boards make up the system. The mouse uses a USB cable to interface with the host PC. On the mouse main PCB, there are three buttons: left, right, and center (scroll wheel). The main board also holds the OvationONS II SoC. Attached to the main board is a daughter card holding four additional buttons, a logo LED, and a multi-color LED that indicates the level of resolution the sensor is set to.

## 1.2.3 Theory of Operation

The mouse incorporates the CYONS2100 42-pin QFN device for controlling buttons, scroll wheel, LED(s), and full speed USB activity.

Power for the mouse is provided via standard USB 5V bus power.

The mouse has a total of seven buttons. The two top buttons control left and right button mouse functionality. There is a center button as well, which is activated by pressing down on the scroll wheel. Other buttons include two buttons for navigating forward and backward between web pages in a browser. Two more buttons, located on the top side of the mouse, control the DPI resolution of the sensor - one to increase resolution, one to decrease resolution. Finally, a scroll wheel is implemented to scroll up and down within software applications.

The mouse is a human interface device (HID) and interfaces with a PC via a five wire full speed capable USB cable.

The CYONS2100 device has 32k of flash storage for programming. 2k bytes of SRAM for data storage are also available on-chip. The main PCB includes a 5-pin USB connector, which can be used for both programming the MCU, and interfacing with the host PC. A MiniProg-to-USB adapter, board number PDC-9241, is needed for programming the OvationONS II device. This is a separate board included in the kit.

A snap-fit lens for the sensor is attached to the main board.

## 1.3 Document Revision History

Table 1-1. Revision History

Revision	PDF Creation Date	Origin of Change	Description of Change
**	11/24/2008	FJZ	New Guide
*A	04/24/2009	FJZ	KIt Updates
*B	04/29/2009	FJZ	Updated Service Pack version and Changed compiler to ImageCraft
*C	06/12/2009	FJZ	RTM Release

## 1.4 Documentation Conventions

Table 1-2. Document Conventions for Guides

Convention	Usage
Courier New	Displays file locations, user entered text, and source code: C:\...\cd\icc\
<i>Italics</i>	Displays file names and reference documentation: Read about the <i>sourcefile.hex</i> file in the <i>PSoC Designer User Guide</i> .
[Bracketed, Bold]	Displays keyboard commands in procedures: [Enter] or [Ctrl] [C]
File > Open	Represents menu paths: File > Open > New Project
<b>Bold</b>	Displays commands, menu paths, and icon names in procedures: Click the <b>File</b> icon and then click <b>Open</b> .
Times New Roman	Displays an equation: $2 + 2 = 4$
Text in gray boxes	Describes Cautions or unique functionality of the product.



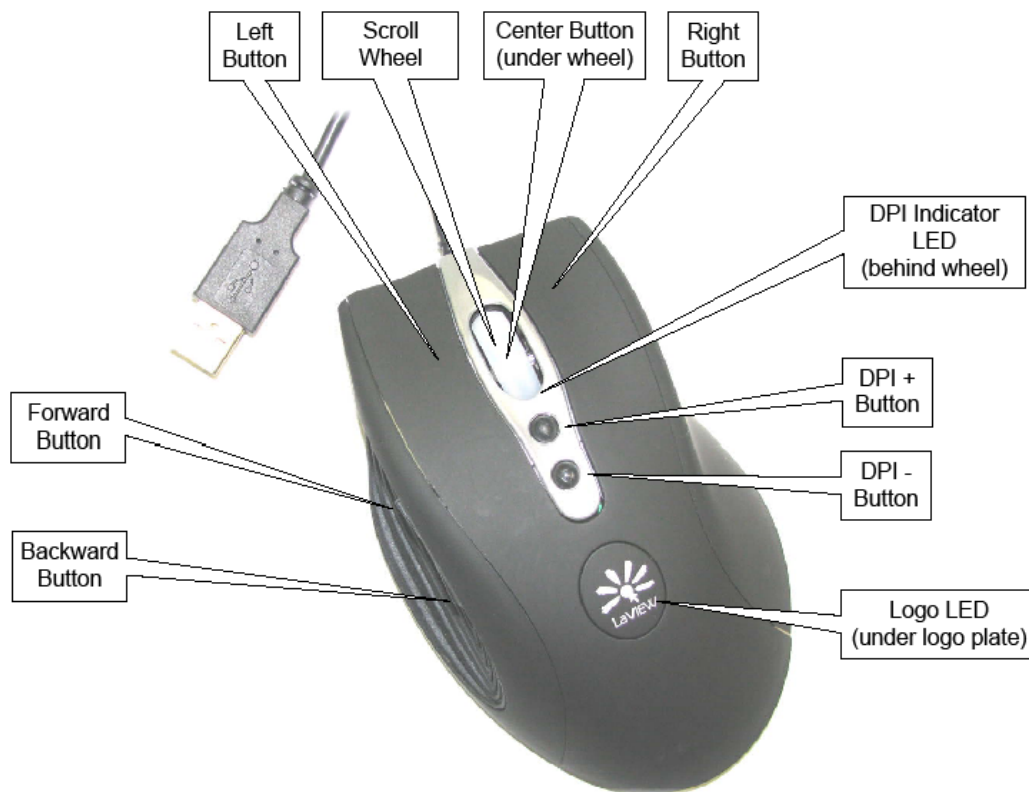


## 2. Hardware



### 2.1 Mouse Overview

The following image shows the major components of the mouse.



### 2.2 Major Components

Major components include the CYONS2100 SoC, push button switches, scroll wheel, bulk, and bypass capacitors, LED(s), and resistors.

### 2.3 Additional Electrical Feature

The board includes test pads for all OvationONS II pins. This allows the designer to access the other GPIO's and features supported by the device, without developing a separate board. In this sense, the CY4631 kit can also be used as a development kit for the CYONS2100-LBXC part.

The board also includes a connector for debugging. Using this connector and the Cypress CY3250-FLEXCABLE, a Cypress **ICE Cube** debugger can be used to perform on-chip debugging. With the

ICE, the developer can step through code, set breakpoints, and monitor variable and register settings during operation.

The ICE cube is available for purchase as part of the CY3215 DVK development kit. This kit, and the CY3250-FLEXCABLE, are available for purchase from [www.cypress.com](http://www.cypress.com).

## 2.4 Mechanical

The main board design includes the OvationONS II sensor, a 5-pin connector used for programming and PC-to-mouse USB interface, three push buttons and a mechanical scroll wheel encoder. The main board has two copper layers.

The daughter card contains four push buttons, a logo illumination LED, a resolution indicator LED, and a connector for interfacing to the main board.

Mechanical design is an important consideration. If the designer is developing his own PCB based on this RDK, care should be taken to meet the mechanical requirements of the OvationONS II SoC. These requirements are described in detail in the application note AN48995, "Mechanical Design Considerations for the CYONS2xxx Laser Navigation SoC". The note is included in the kit CD.

## 2.5 HW Documentation

The following documents are included as part of the kit.

- Bill of Materials
- Schematics in PDF and Orcad source format
- PCB layout in PDF, gerber, and Allegro/PADs format

## 2.6 Manufacturing Considerations

As a guide to assist in production of mice, Cypress has published application note AN49035, "Manufacturing Considerations for the OvationONS™ II Laser Navigation System-on-Chip". The note is included in the kit CD.

## 3. Firmware



### 3.1 Introduction

This document covers the design goals, architecture, configuration options, and firmware source code modules for the CY4631 OvationONS II USB Mouse RDK. This document assumes the reader is familiar with the PSoC Designer™ Integrated Development Environment (IDE) software, in addition to the general operation of USB HID devices interface.

An introduction to the PSoC Designer tools is available at the Cypress website [www.cypress.com/psocdesigner](http://www.cypress.com/psocdesigner). The site has a link to a training page which includes Webinars, OnDemand Training, and workshops.

### 3.2 Development Environment

The following tools were used to develop the RDK mouse firmware and can be purchased from Cypress Semiconductor through [www.cypress.com](http://www.cypress.com)

- PSoC Designer™ IDE Software version 5.0, Service Pack 4.5 (included in kit)
- PSoC ICE-CUBE™
- ImageCraft Compiler

### 3.3 Design Goals

#### 3.3.1 Mouse Application

The RDK firmware design includes configuration options to

- Target firmware builds to adjust for sensor orientation and CPI adjustment. (See *global.h* and *SVCE\_Cpi()* service function)
- Support different features for all OvationONS II Optical Sensors
- Support for left, middle, right, scroll wheel, forward, back, CPI up, and CPI down buttons
- Support for USB remote wakeup on any button press, scroll wheel movement, or mouse movement
- Debounce all button presses
- Support 1ms USB report rate

#### 3.3.2 USB Functionality

The RDK firmware uses the full-speed USB (FSUSB) User Module in PSoC Designer.

- The USB User Module provides the USB functionality required for the RDK.
- The RDK is designed to be attached to any USB host interface.
- The USB User Module is designed to handle 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 USB User Module allows customization of both the standard USB Descriptors and USB HID Report Descriptors.

### 3.3.3 Bootloader Functionality

The bootloader enables end-user upgrades of the mouse's operating firmware. A typical upgrade operation requires the following steps:

- Preparation of a download file that contains the new firmware. Typically this is provided by the mouse manufacturer, and offered as an upgrade to end users on the manufacturer's website.
- Preparation of the mouse to accept the download file. This is referred to as "invoking the bootloader". In this step, the user performs a prescribed action on the mouse, which
  - Suspends normal operation of the mouse
  - Breaks the USB connection between the host PC and the mouse
  - Restarts the mouse as a bootloader-compatible USB device
- Launch of a PC-based application that takes the upgrade download, and sends the new firmware to the mouse over the bootloader USB interface.
- Restart of the mouse with upgraded firmware.

The CY4631 RDK's firmware invokes the bootloader when it detects simultaneous presses of the left, right, middle, CPI increase, and CPI decrease buttons. Upon this key sequence, the device stops operating as a mouse, and restarts as a bootloader compatible device. At this time, it is ready to accept a download file from the bootloader host application.

## 3.4 Architecture

The firmware architecture consists of PSoC Designer User Modules plus application code.

### 3.4.1 User Modules

#### 3.4.1.1 *Full-Speed USB Bootloader User Module*

The bootloader user module (UM) provides firmware updates and client-driven replacement of firmware when the user invokes the bootloader. It also provides USB interface between the host and the RDK application. The functionality provided by the bootloader UM includes:

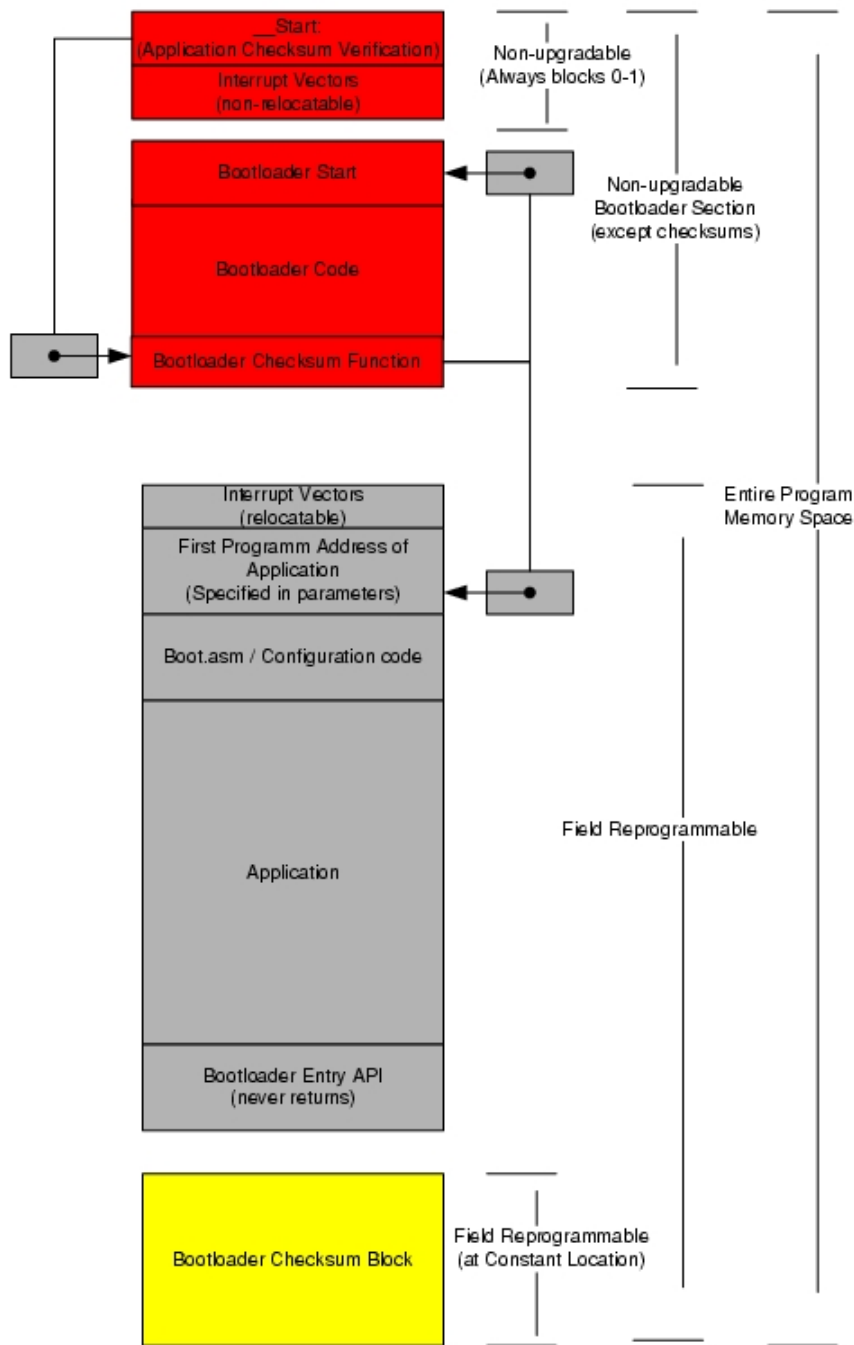
- USB device interface driver
- USB Chapter 9 compliance
- Mouse HID class support
- Setup wizard for USB descriptor generation and modification
- Standard APIs for programmatic control of the USB UM

#### **Bootloader User Module Parameters**

Parameters must be set so that there is sufficient EEPROM space and no EEPROM space conflicts between application firmware, bootloader firmware, checksum block, and any EEPROM UM blocks. If the designer adds new code to the project, it may be necessary to increase the size of the bootloader block. Refer to the user module datasheet in PSoC Designer for more information.

The bootloader host application checks the bootloader key which is user defined to any value.

Properties - BootLdrUSBFS	
Name	BootLdrUSBFS
User Module	BootLdrUSBFS
Version	1.2
ONE_Block_Relocatable_In	0x2D
ApplicationCode_Start_Block	0x2F
Last_Application_Block	0xFE
Application_Checksum_Bloc	0xFF
Bootloader_Start_Block	0x2
Bootloader_Size	0x2B
BootLoaderKey	0001020304050607
Flash_Program_Temperature	20C
ICE_Debug_FLASH_DISABLE	Flash_WRITE_ENABLE
BootLdrUSBFS_ver	0x1000
Bootload_when_CKSUM_fail	DISABLE_(development)



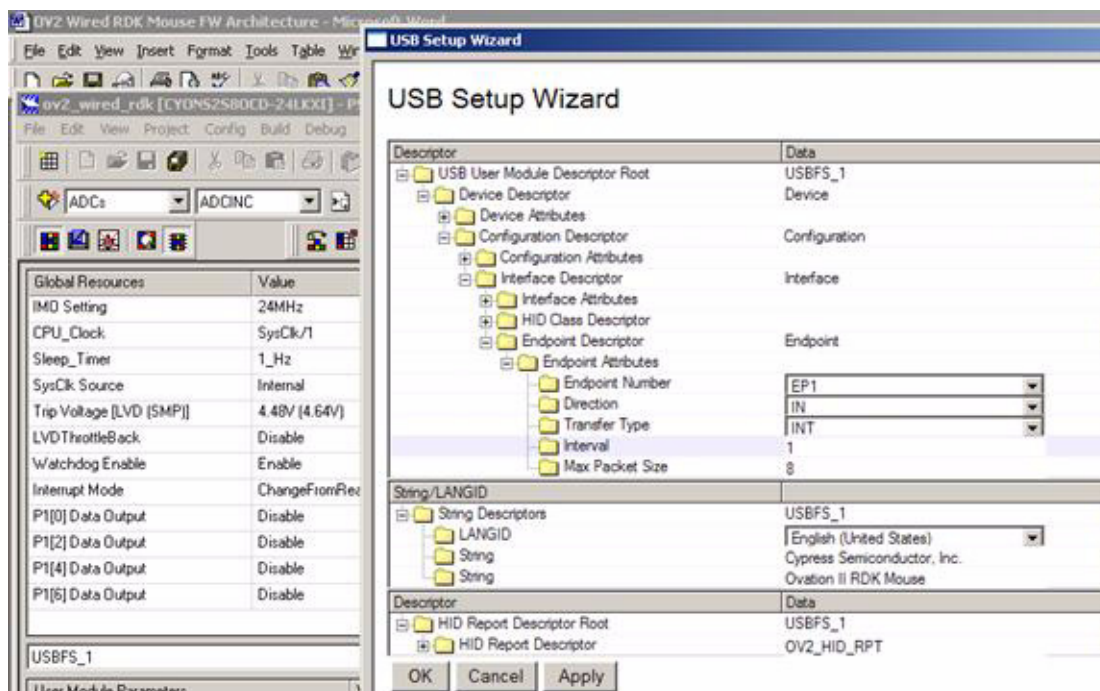
For detailed information on setting up the bootloader UM parameters, open the **Protocols** folder in the User Module Catalog, then right-click on the **BootLdrFSUSB** folder, and select **Datasheet**.

## Full Speed USB Bootloader Setup

The bootloader user module can be configured by using the setup wizard in PSoC Designer 5.0. The bootloader user module consists of two USB interfaces: the normal mouse application, and the bootloader application. Only one interface is active at any time.

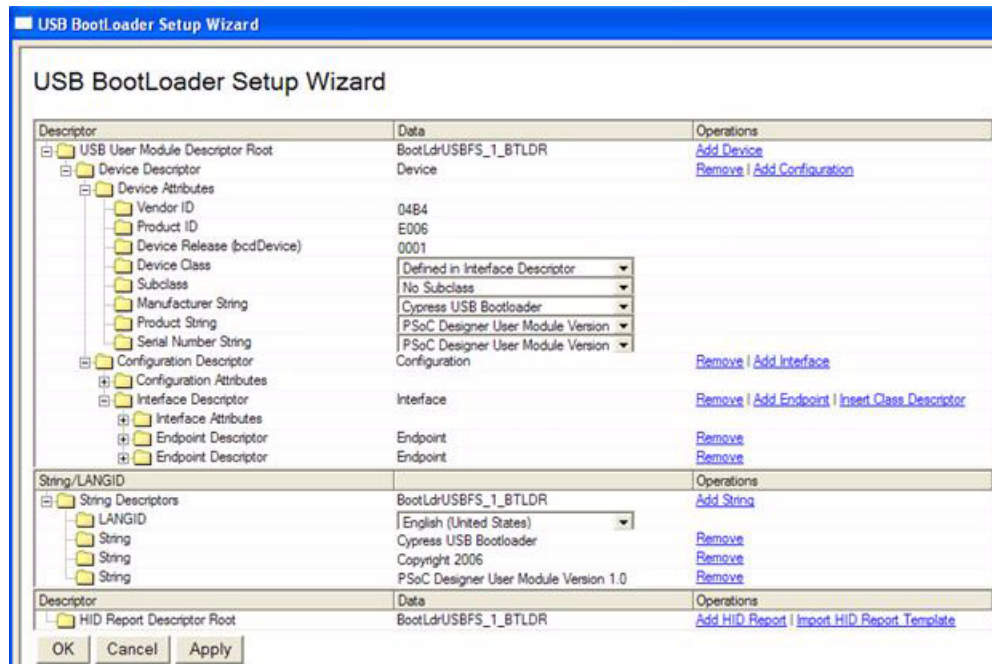
Both interfaces can be changed by running the setup wizard. Typical changes include changing the USB polling interval and modifying the USB VID/PID.

To change the USB polling interval, change the HID report rate by updating the Interval descriptor for EP1. Go to device editor in PSoC Designer, right-click the USB user module, and select the Device: Application USB Setup Wizard. Change the interval to 1 to let the USB host know to poll every 1ms for a USB report. This results in the highest report rate possible.



The device has two USB interfaces. Typically, boot.asm presents the application firmware interface when there is no need to execute the bootloader function. If the bootloader is invoked, then bootloader code presents the bootloader USB interface to the USB host PC. To set your device's PID/VID for both interfaces, enter your PID/VID by selecting Device: Bootloader USB Setup Wizard and Device: Application USB Setup Wizard.

Figure 3-1. USB BootLoader Setup Wizard



#### 3.4.1.2 EEPROM User Module

The EEPROM UM provides memory of user CPI settings and Logo LED on/off setting. EEPROM UM uses the following device editor settings:

- Module name = E2PROMx128
- 1st Block = 1
- Length = 128

The first block is the starting block in which the EEPROM device resides for the specific device. Valid start block locations range from 0 to 511 for 32K devices, 255 for 16K devices, 127 for 8K devices, 63 for 4K devices, and 31 for 2K devices.

#### 3.4.1.3 NAV User Module

The NAV UM provides OvationONS II device initialization and tracking support:

- Initializes the sensor, based on the device ID in which is factory-programmed into the Ovation device.
- Sets individual sensor settings based on calibration data stored in the device.
- Starts or stops laser and tracking in OvationONS II.
- Reads tracking count changes in X and Y directions.
- Sets CPI resolution.

NAV UM uses the following device editor settings: (Note that lift detection support is not yet available in firmware.)

- User Module Name = NAV\_UM
- Navigation Sleep Modes = Enable



### 3.4.1.4 *Timer16 User Module*

The Timer UM provides a one-millisecond timer ISR routine. The application uses this timer and several counters to time-out the USB connection and allow continuous pressing of the CPI buttons to adjust CPI.

The Timer16 UM uses the following device editor settings:

- User Module Name = TickTimer
- Period = 32
- Mode = Continuous
- Clock Select = 32 kHz

These settings produce a continuous, 1 ms timer interrupt.

### 3.4.2 *Application Code*

The CY4631 OvationONS II USB Mouse RDK application, besides being the code that ties the User Modules together, handles the following tasks:

- Initialization and configuration of all blocks in the design
- Execution of the polling loop in main.c
- Control of USB idle state
- Control of the display LEDs
- Polling of buttons, scroll wheel, and optical sensor
- Debounce support for button presses
- Control of USB suspend, resume and remote wakeup
- Generation and USB transfer of the mouse reports to the host

This section describes the application code in detail, covering each global data structure and firmware source code file in the reference design.

#### 3.4.2.1 *Global Data Structures*

##### **Persistent Parameters Table**

This structure is implemented over FLASH "EEPROM," read into RAM on system initialization, and contains user defaults that must survive power off/on cycles.

```
typedef struct STRUCT_PERSISTENT_PARAMETERS
{
    UINT16  wCheckWord;

    // User-Selected Defaults Section
    UINT16  uCPI;
    BOOL    fLogoOn;
}
```

## HID Report Structure

This structure contains the Human Interface Device (HID) data report sent by USB packet to the USB host. Sub-structures `BUTTON_REPORT` and `WHEEL_REPORT` contain right button, middle button, left button, and wheel states. `OPTICAL_REPORT` contains 32 bits of X and Y changes read from the OvationONS II device.

```
typedef struct MOUSE_HID_REPORT
{
    BUTTON_REPORT    SButtons;
    OPTICAL_REPORT   SxyData;
    WHEEL_REPORT     SWheel;
} MOUSE_HID_REPORT, *pMOUSE_HID_REPORT;
```

## Wheel Report Structure

This one-byte structure contains the encoder state to sense and report changes in wheel motion.

```
typedef struct
{
    INT8 cz;
} WHEEL_REPORT;
```

## Optical Report Structure

The x/y data is held in a structure consisting of two bytes each for x and y.

```
typedef struct OPTICAL_REPORT
{
    UINT8 bXLow;
    UINT8 bXHigh;

    UINT8 bYLow;
    UINT8 bYHigh;
} OPTICAL_REPORT;
```

## Button Report Structure

The button data is held in a single byte, with each bit corresponding to one of the mouse buttons. The upper three bits are undefined, and the remaining five bits (in MSB to LSB order) are FWD button, BACK button, MIDDLE button, RIGHT button, and LEFT button.

```
typedef struct
{
    UINT8 bfHidMask;
} BUTTON_REPORT;
```

### 3.4.2.2 Source Code Files

This section covers key source code files. Other source code files are covered in the function descriptions that follow.

#### **boot.tpl**

The *boot.tpl* file contains user-defined constraints for PSoC Designer when it generates the boot.asm file for your project. In the CY4631 firmware project, there are five customizations to *boot.tpl*. All changes are to interrupt processing, so that the correct interrupt service routine is called.

*boot.tpl* lines 300 - 303:

```
;;INT_VEC5: equ  RELOC_INT_VECTORS + 14h                ;GPIO Interrupt Vector
    org INT_VEC5
    ljmp _GPIO_ButtonWheelIsr
    reti
```

Lines 407 - 412:

```
;;INT_VEC22: equ  RELOC_INT_VECTORS + 58h                ;USB_Wakeup
    org INT_VEC22
export USB_WAKEUP_USER_ISR_VECTOR
USB_WAKEUP_USER_ISR_VECTOR:
    ljmp _USB_WakeUpIsr
    reti
```

Lines 419 - 422:

```
INT_VEC24: equ  RELOC_INT_VECTORS + 60h                ;;Sleep Ovation2
    org INT_VEC24
    ljmp _NAV_SleepIsr
    reti
```

Lines 424 - 427

```
INT_VEC25: equ  RELOC_INT_VECTORS + 64h                ;Wakeup Ovation2
    org INT_VEC25
    ljmp _NAV_WakeIsr
    reti
```

Lines 430 - 434

```
GENERIC_VEC26: equ  (RELOC_INT_VECTORS + 68h)          ;Ovation2 Global Interrupt
    org INT_VEC26
;`@INTERRUPT_26`
    ljmp _NAV_GlobalIsr
    reti
```

### flashsecurity.txt

The flash security file allows EEPROM read and write for storing persistent data, such as user CPI settings, and bootloader FW updates. The flash security file is configured to allow write access to block 4 of flash, so that persistent data can be modified by the user. *Flashsecurity.txt* marks all EEPROM locations with a "u" for "unprotect" to allow firmware read/write access.

```
; 0 80 100 180 200 280 300 380 (+) Base Address
   u  u  u   u   u   u   u   u ;   Base Address 0
```

#### 3.4.2.3 Source Code Functions

This section describes the various functions in the application code.

##### Main Function

```
void main(void)
```

The main c function. It calls the initialization functions for the various blocks, and then runs a loop checking for mouse activity or the need to enter USB suspend mode.

##### Initialization Functions

```
void SYSINI_Button(void)
```

Clears button states to "Not Down" in HID report, disables button interrupts until use in waking up mouse, and turns on the pull-up drive for button pins.

```
void SYSINI_Eeprom(void)
```

Reads persistent parameters from EEPROM. Sets up default parameters if no previous EEPROM data found.

```
void SYSINI_Nav(void)
```

Starts DSP, laser, and sets default tracking CPI. It primarily calls OvationONS II NAV User Modules.

```
void SYSINI_Led(void)
```

Calls SVCE\_LED. Sets initial LED settings for lighting the scroll wheel to indicate CPI level.

```
void SYSINI_Timer(void)
```

Starts the Timer16 User Module.

```
void SYSINI_UsbHidReport(void)
```

Activates USB by User Module, waits until configuration is complete, establish data toggle, and disable USB wake-up ISR.

```
void SYSINI_ZWheel(void)
```

Initialize wheel states (all 0), temporarily mask wheel interrupts, turn wheel pull-ups on.

### Service Functions

```
void SVCE_Buttons(void)
```

Reads and debounces all buttons.

```
void SVCE_Cpi(void)
```

Adjust CPI up or down depending on fCpiUp and fCpiDown flags. Flags are set by TASK\_ButtonsIsr().

```
void SVCE_Eeprom(void)
```

Provides write of persistent parameters to EEPROM. In this case, user CPI adjustments.

```
void SVCE_HidReport(void)
```

Polls for XY tracking counts, wheel, and button state changes. Packages and posts a HID report over USB protocol, if necessary.

```
void SVCE_Led(void)
```

Sets initial LED settings for lighting the scroll wheel to indicate CPI level.

```
void SVCE_UsbResume(void)
```

Resume USB from suspend state. If no remote wakeup, then turn on track init and start laser.

```
void SVCE_UsbSuspend(void)
```

Suspend USB with or without remote wakeup.

```
void SVCE_WheelDetect(void)
```

Reads and updates the state of the zWheel.

### ISR Functions

```
void GPIO_ButtonWheelIsr(void)
```

When buttons are pressed, this ISR sets fHaveButtonInt to TRUE allowing firmware to exit sleep and send wake-up signals to a host PC. Host PC should have remote wake-up enabled.

```
void NAV_SleepIsr(void)
```

Sets the fNAVSleepInt to TRUE. Detecting the TRUE condition instructs firmware to enter or reenter sleep mode.

```
void USB_WakeUpIsr(void)
```

Sets the fUSBWakeInt to TRUE. The RDK mouse firmware uses this flag to detect USB wake-up and exits suspend mode.

## Other Functions

```
void DelayMsecs(UINT16 bDelayMsecs)
```

A general purpose timing function that delays for the indicated number of milliseconds.

```
void ButtonPortDriveOn(void)
```

Sets the drive mode for the GPIOs connected to the button switches.

```
void ButtonPortDriveOff(void)
```

Sets the button GPIO's to a low state.

```
void LogoLedPortDriveOn(void)
```

Turns on the logo LED.

```
void LogoLedPortDriveOff(void)
```

Turns off the log LED.

```
void ColorLedPortDriveOn(UINT8 bfColorLedBits)
```

Turns on the LED that indicates the CPI setting. The implementation supports tri-color LED in either a common-anode or common-cathode configuration.

```
void ColorLedPortDriveOff(void)
```

Turns off the LED that indicates the CPI setting. The implementation supports tri-color LED in either a common-anode or common-cathode configuration.

```
void PreResetWarningLedLoop(void)
```

An LED blinking loop used during resets.

```
void FatalLogoLedLoop(void)
```

An LED blinking loop to be used to indicate failures. This function never exits.

## 3.5 Example Project Build

This section provides step-by-step instructions for building a project using the CY4631 RDK Mouse with PSoC Designer 5.0. This shows the basic use of the PSoC Designer and PSoC Programmer tools, and how to make basic changes to the firmware and operation of the mouse.

### 3.5.1 Project Change

For this example, assume that the designer wishes to change the behavior of the logo LED. In normal operation, the logo LED is illuminated when the mouse is in motion. In this example, the firmware is changed so that the logo LED is on at all times.

### 3.5.2 Step-By-Step Changes

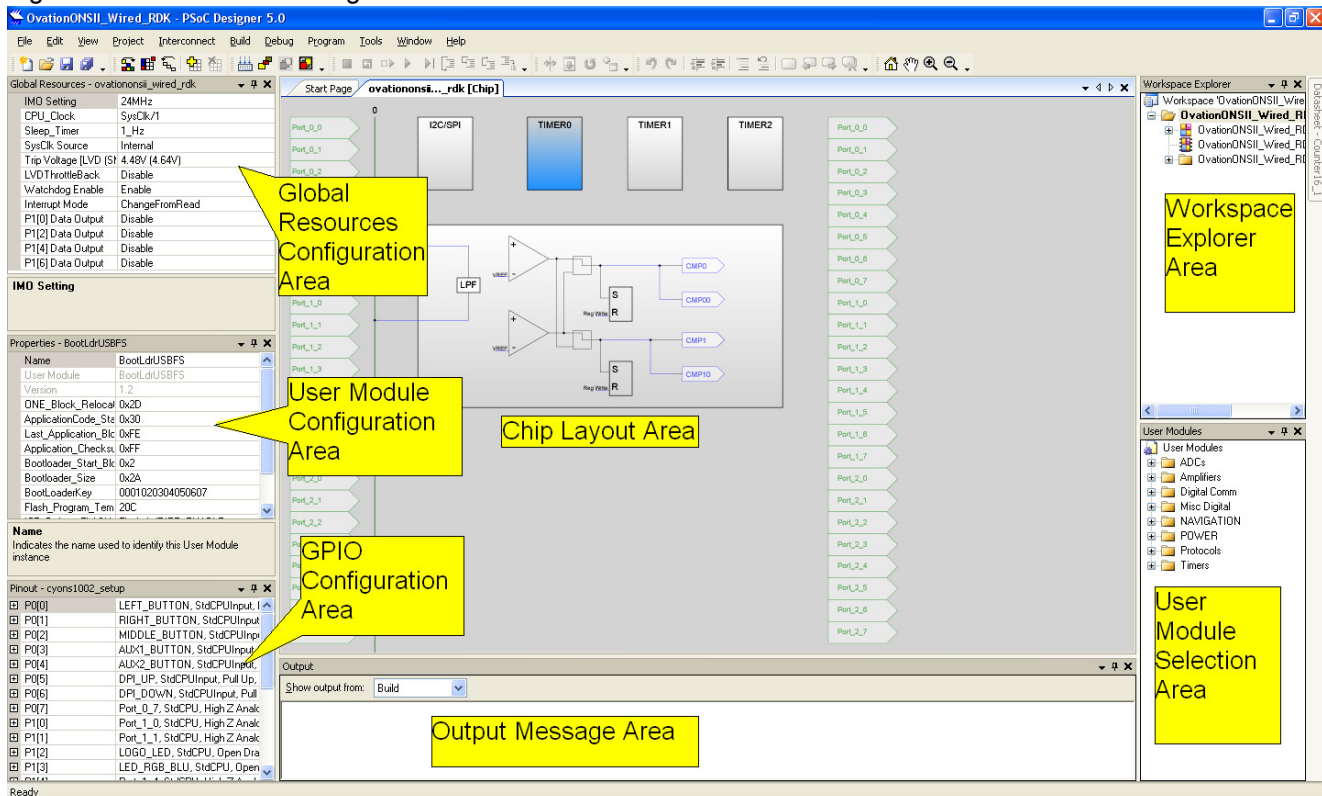
#### 3.5.2.1 Step 1 - Launch PSoC Designer

The first step is to open the project in PSoC Designer by double-clicking the *OvationONSII\_Wired\_RDK.app* file in the project's directory. After PSoC Designer is launched, the main screen can be seen.

The areas of the PSoC Designer screen are as follows:

- Global Resource Configuration Area - This area allows selections of global parameters of the device, such as clock speed,
- User Module Configuration Area - This area allows changes to the configuration settings of each user module. The active user module in this window is selected in the User Module Selection Area.
- GPIO Configuration Area - This area is used to configure the GPIO's, including the GPIO name, drive mode, and interrupt enable/disable.
- Chip Layout Area - This area shows the layout of the chip, showing the hardware-based interconnection among the GPIO's, mux buses, and user module blocks.
- Workspace Explorer Area - This area allows the user to navigate to all the files in the solution workspace, including source files, library files, output files, and user module documentation.
- User Module Selection Area - This area allows User Modules to be selected, named, and placed in the design.
- Output Message Area - This text area gives results of any actions taken by the user, such as error messages that may result from a project build.

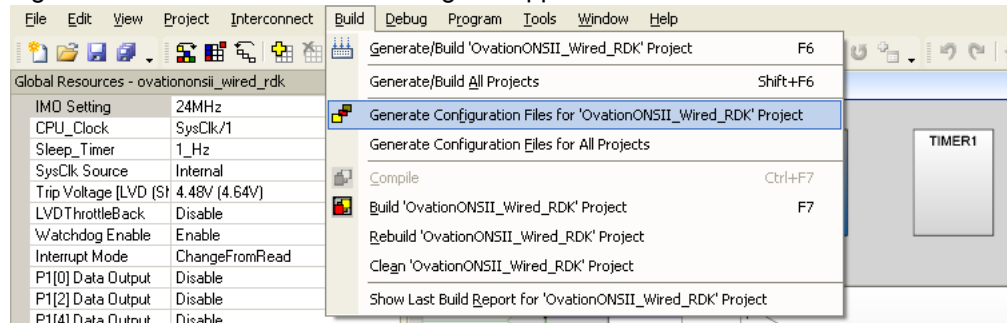
Figure 3-2. Main PSoC Designer Screen



### 3.5.2.2 Step 2 - Generate the Application

The next step is to generate the application, by selecting **Generate Configuration Files** under the **Build** menu item, or by clicking on the **Generate Configuration Files** icon. This creates the required library source files based on the configuration settings and user module selections.

Figure 3-3. Menu Item for Generating the Application

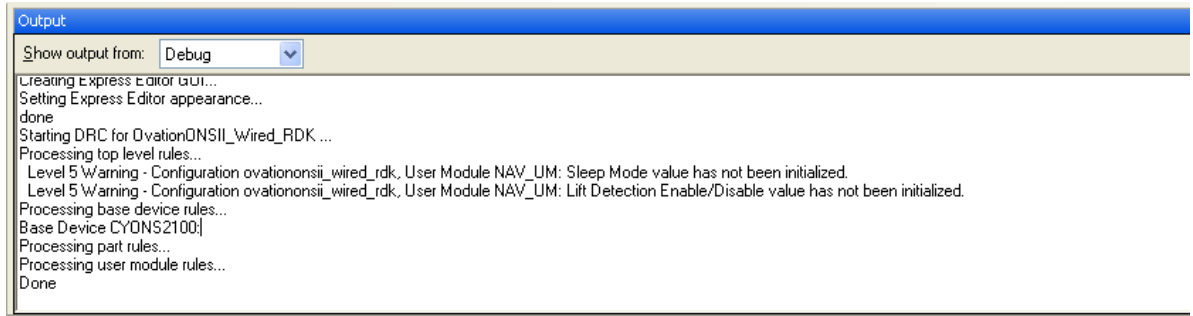


### 3.5.2.3 Step 3 - Check for Errors

Next, check the Output Message Area to confirm that there were no errors resulting from the previous step.

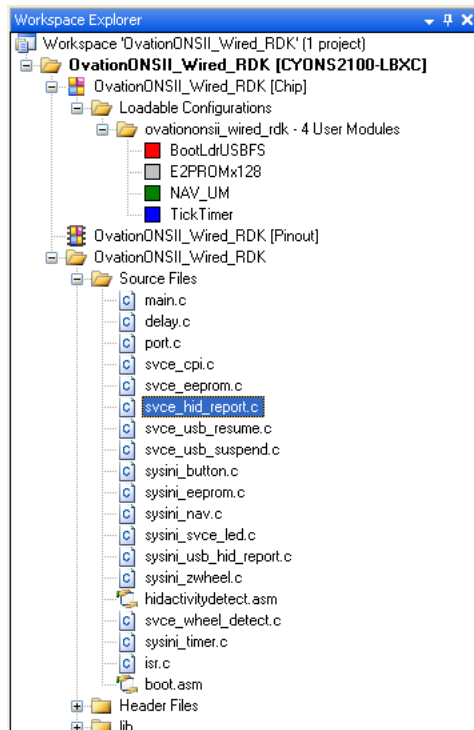


Figure 3-4. Checking the Output Window for Errors



### 3.5.2.4 Step 4 - Open the Appropriate Source File

The next step is to make the desired firmware change. This is accomplished by opening the source file `svce_hid_report.c`, by double-clicking the file name in the expanded Workspace Explorer Area.

 Figure 3-5. Opening `svce_hid_report.c`


### 3.5.2.5 Step 5 - Edit the Source File

To make the change to the logo LED, just comment out the line of code that turns it off. In the version below, this is accomplished by commenting out line 147.

Comment out line 147 to leave the logo LED on

```

//Zero out accumulated report data for new data
bAccumulateWheel = 0;

uTickACKTimer = USB_HOST_ACK_TIMEOUT;

}

}

else
{
    // need timeout check for normal S3 resume (CDT 31908) where host may take a while to
    // toggling the USB EP until we are back in sync.
    if(uTickACKTimer == HOST_TIME_OUT)
    {
        USBSendNullReport();
    }
}

//LogoLedPortDriveOff();
DebugWarning();

}

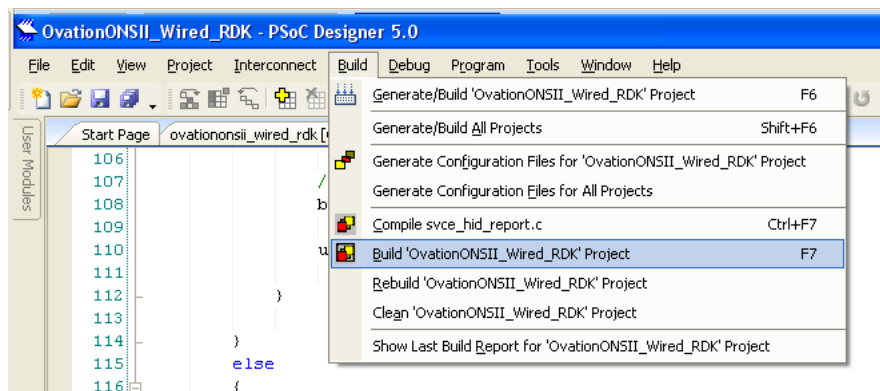
void USBSendNullReport ()

```

### 3.5.2.6 Step 6 - Build the Solution

Next, build the solution by selecting **Build** from the **Build** menu, or by directly clicking the Build icon. As before, check the Output Message Area for any errors.

Figure 3-6. Build the Solution

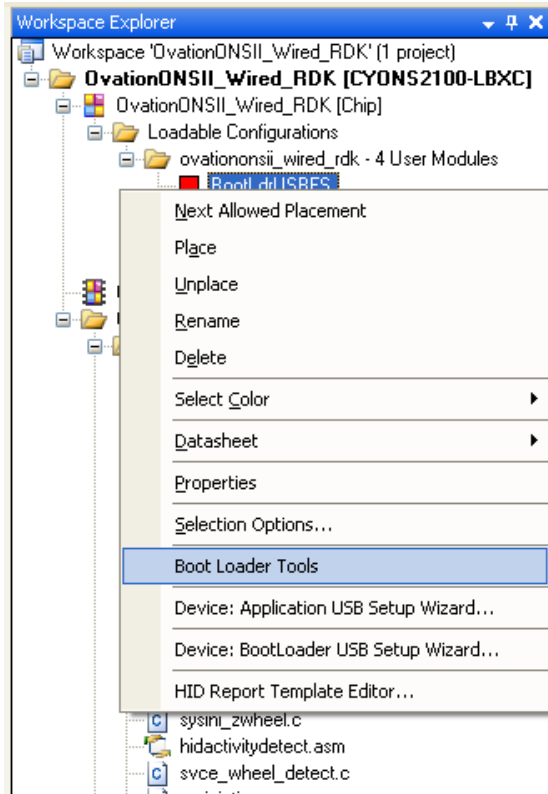


### 3.5.2.7 Step 7 - Launch Bootloader Tools

One complication of the bootloader function is that two checksums must be recorded. The first is the overall checksum of the entire firmware image. The second is the checksum of the application portion of the firmware. Because the overall checksum is affected by the application checksum, the project must be built twice. In the second build, the application checksum is known, and used to generate the overall checksum.

Tools for the second build are available as **Boot Loader Tools**, which is accessible from the Workspace Explorer Area. Right-click the bootloader user module, and select **Boot Loader Tools** to open the bootloader tools window.

Figure 3-7. Launching the Bootloader Tools



### 3.5.2.8 Step 8 - Rebuild Project with Valid Checksum

Click on the **Rebuild Project with Valid Check Sum** button to perform the second build. When complete, there should be a green message indicating success.

Figure 3-8. Rebuilding with Valid Checksum

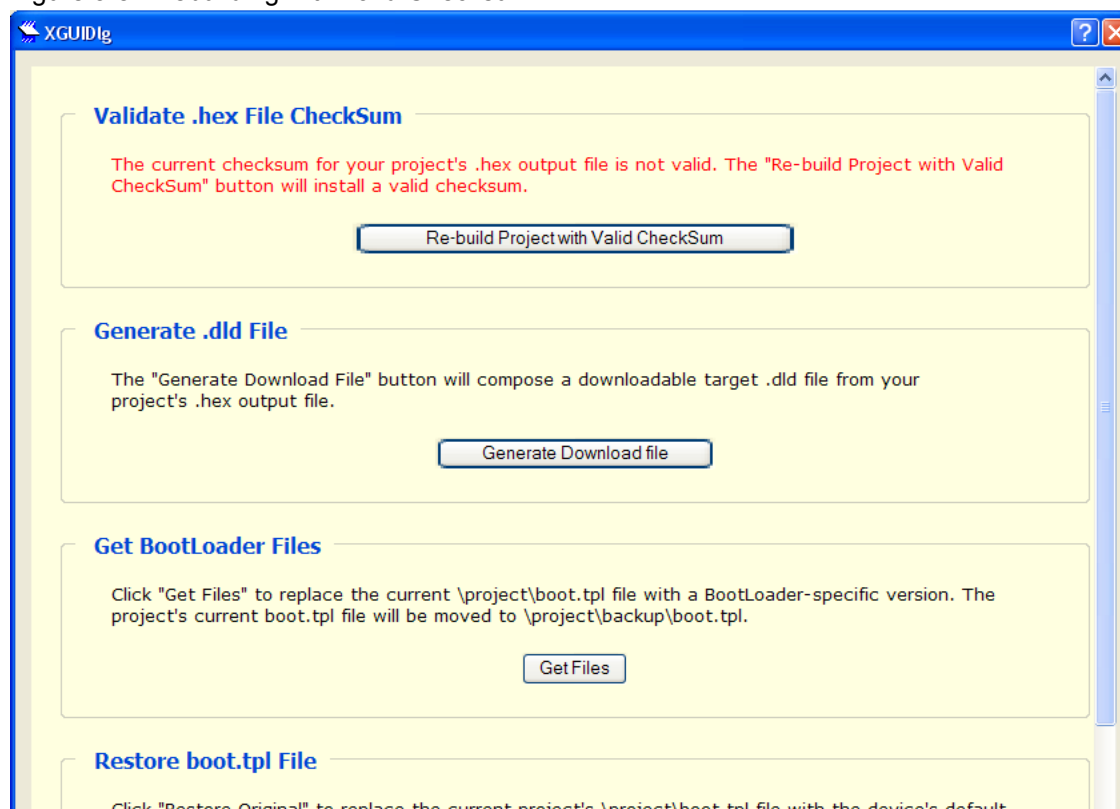
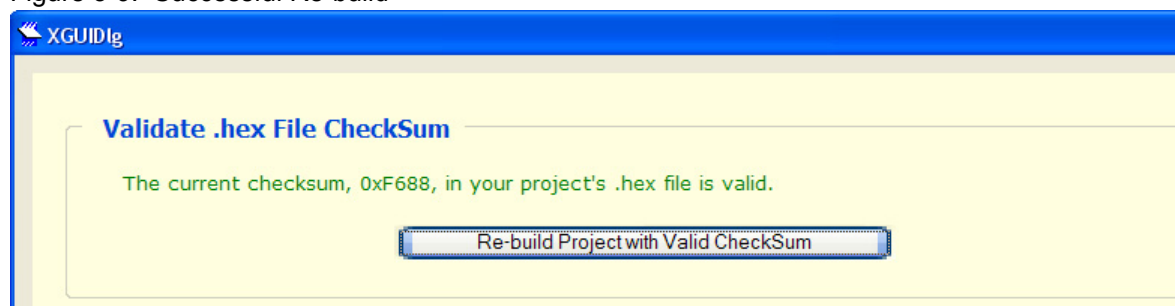


Figure 3-9. Successful Re-build



### 3.5.2.9 Step 9 - Prepare to Program the Device

The next step is to download the new code into the CYONS2100 device. First, set up the programming hardware:

- Plug the Cypress MiniProg into a USB port
- Plug the CY3655-PLG adapter board into the MiniProg. Make sure that the -PLG's jumpers are set to the "E3" (Full-Speed USB) position, by connecting J3 pin 1 to pin 2, and pin 3 to pin 4.
- Plug the CY4631 RDK Mouse into the other side of the CY3655-PLG

Next, launch PSoC Programmer by selecting **PSoC Programmer** from the **Program** menu item.

Figure 3-10. Launching PSoC Programmer

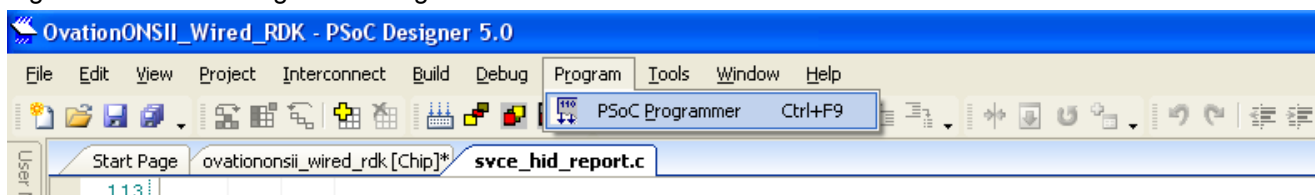
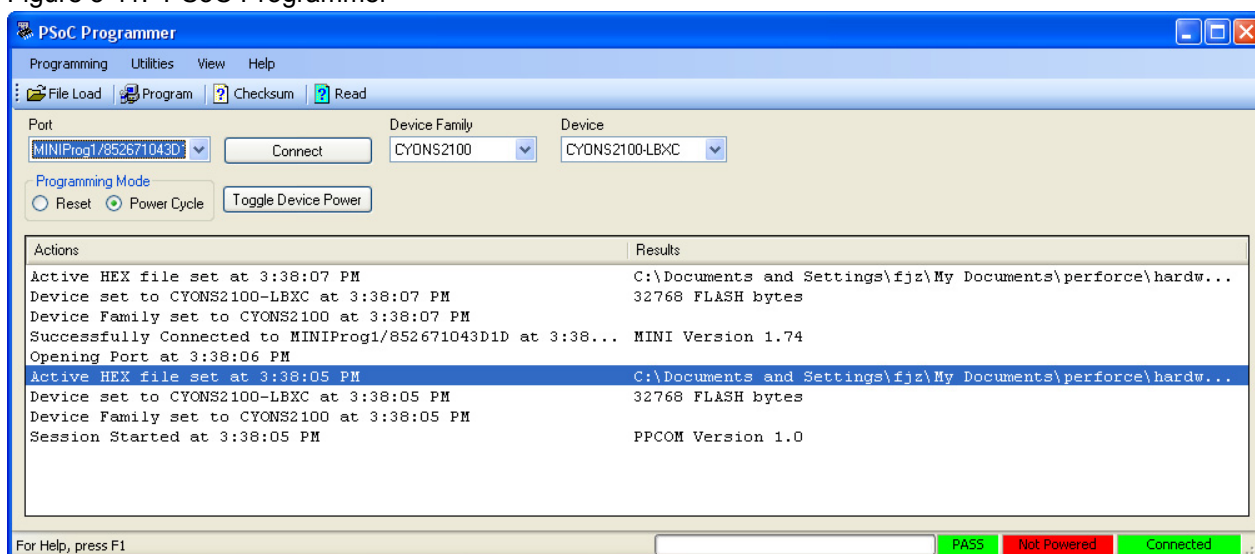


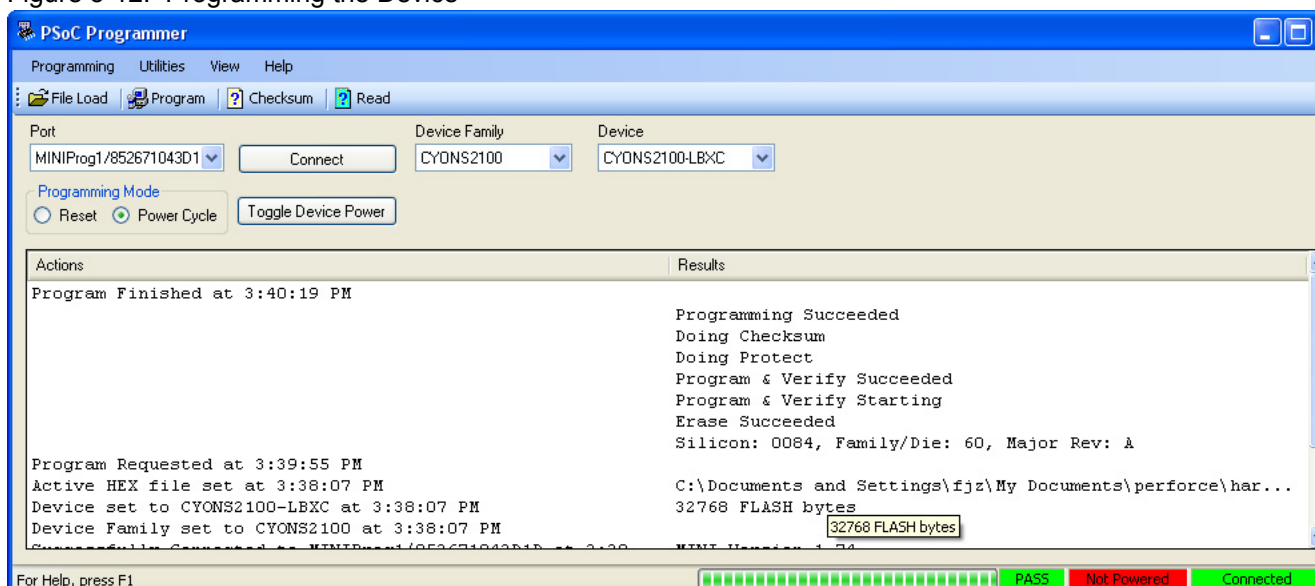
Figure 3-11. PSoC Programmer



### 3.5.2.10 Step 10 - Program the Device

Before programming the device, make sure the firmware version of the MiniProg is 1.74 or later. If needed, upgrade the MiniProg firmware by selecting **Utilities**, then **Upgrade Firmware**. When complete, click the **Program** button, and wait for programming, and verify that the programming succeeds.

Figure 3-12. Programming the Device



#### 3.5.2.11 *Step 11 - Test*

Remove the RDK Mouse from the CY3655-PLG and plug it into the PC. The logo LED should stay lit at all times. At this point, you have successfully modified the RDK mouse firmware.

## 4. Regulatory Tests



### 4.1 Introduction

The CY4631 Wired Mouse Reference Design has been successfully tested for compliance with several regulatory and standards body requirements. This section lists the tests and results of those tests.

### 4.2 USB

The CY4631 was successfully tested as a full-speed device against the following USB specifications.

- USB 2.0, Chapter 7
- USB 2.0, Chapter 8
- USB 2.0, Chapter 9
- HID Class Specification version 1.1

### 4.3 FCC

The CY4631 was successfully for radiated and conducted emissions against the requirements of FCC Part 15, Subclass B, for a class B digital device.

### 4.4 ESD

The CY4631 was successfully for ESD immunity using standard EN55024, up to a level of 8 kV.

### 4.5 WLK

The CY4631 successfully passed Windows Logo Kit testing on SMSC and Cypress based hubs. Other hubs exhibited failures with full-speed mice. If you are experiencing failures during WLK testing, do contact Cypress Applications, and reference “CDT32379”.

