AN57473

USB HID Basics with PSoC® 3 and PSoC 5LP

Author: Robert Murphy
Associated Project: Yes
Associated Part Family: All PSoC 3 and PSoC 5LP parts
Software Version: PSoC Creator™ 3.3 and higher
Related Application Notes: See Additional Resources

AN57473 describes the basics of the USB Human Interface Device (HID) protocol, and how to implement it in PSoC® 3 and PSoC 5LP. It explains how to configure USB input transactions using the PSoC Creator™ USBFS Component, with basic mouse and joystick inputs as examples. This application note is a prerequisite for the intermediate-level AN58726.

Contents

1 Introduction

USB is a complex protocol, and it can be difficult for beginners to quickly get a USB-based application up and running. However, some aspects of USB are easy to use, especially the human interface device (HID) protocol. HID is designed for common PC interface devices such as keyboard and mouse, but can be adapted for many custom applications. Most PC operating systems, including Windows, Mac, and Linux, include HID drivers. This means that you don't have to write a driver, instead you can focus on developing your application firmware.

This application note shows you how to do simple data transfer with PSoC 3 and PSoC 5LP using USB HID. The PSoC devices include a dedicated Full-speed (FS) 12-Mbps USB 2.0 peripheral, which uses an internal oscillator – a crystal is not required. The PSoC Creator IDE and the USB Full-Speed (USBFS) Component make it possible to quickly build an application. A mouse and a joystick are used as examples.

This application note assumes that you are familiar with developing applications using PSoC Creator for PSoC 3 or PSoC 5LP. If you are new to these products, see AN54181, Getting Started with PSoC 3 and AN77759, Getting Started with PSoC 5LP. If you are new to PSoC Creator, see the PSoC Creator home page.

If you are new to USB, basic concepts are explained in AN57294, USB 101: An Introduction to Universal Serial Bus 2.0. If you are familiar with USB basics, more advanced level information is given in Related Application Notes.

2 Introduction to HID

Human interface devices (HIDs) are used daily in both our personal and professional lives. They are the keyboard and mouse attached to computers, the gaming controller used to play video games, or the digital pen / tablet used to draw. As the name suggests, it is a device that creates an interface between humans and computers. It also includes some devices that you may not normally think of as falling under the HID specification, such as a bar code scanner. Figure 1 shows examples of various HIDs.

Figure 1. Common Examples of HIDs

When you plug any USB device into a host PC, the PC requests information about that device. This information is presented to the PC in the form of descriptor tables, or descriptors. There are many types of descriptors; two are important for HIDs:

- Interface descriptor: defines the USB device as an HID.
- Report descriptor: defines the format and usage of the data that the device provides. For example, a USB mouse reports data on X and Y movement and button activity, and its report descriptor defines the structure and format of that data.

One important feature of HID report descriptors is that there are hundreds of ways to set up and organize them to define the same device. The report descriptor serves many functions, for example:

- Provide information about device features
- Specify the organization of the device's data
- Identify if the data has an exponential value associated with it (such as 10^{-3}, 10^{3})
- Specify units associated with the data

Depending on the device, report descriptors can be very simple or highly detailed and complex. However, once you understand descriptor essentials you can easily develop your own descriptors.

3 Report Descriptor Details

Report descriptors are made up of many items; an item is a distinct grouping of data. An item can be of type:

- Main
- Global
- Local

It is important to distinguish between an item type and the item itself. Under the three item types listed above, there are many different items. Table 1 lays out the items that are discussed in this application note. Because report descriptors can get very complicated, this application note focuses on the more common items. Many additional items can be found in the USB HID Specification and the HID Usage Tables available from the USB Implementers Forum (http://www.usb.org). Both these documents contain more detailed information about the item types and items discussed.

Table 1. Common Items

Item Type	Items	Item Category
Main	Input	
	Output	
	Feature	
	Collection	Physical
		Application
		Logical
Global	Usage Page	
	Logical Max and Min	
	Report Count	
	Report Size	
Local	Usage	
	Usage Max and Min	

3.1 Main Items

Main items are used either to define what the data contains or to group data together. There are five different items of type Main:

- Input
- Output
- Feature
- Collection
- End Collection

Input, Output, and Feature types are used to define items while Collections and End Collections are used for grouping purposes. An **Input** item refers to the data that the device sends to the host (such as a mouse button click), while an **Output** item refers to data that the host sends to the device (such as the Caps Lock LED on your keyboard).

A **Feature** item is information that the host can both send to the device and read from the device. Feature items contain device configuration information – altering this information changes the behavior of the device. Feature reports are commonly used with PC interface applications where clicking a button on the GUI alters the operation of the device.

Collection items are used to group together Input, Output, and Feature items. In the HID specification, there are three different types of predefined collections:

- Physical
- Application
- Logical

A **Physical** collection consists of data collected at one geometric point. A device that collects data from multiple sensors at one time may group the data in a physical collection. An example of a device that uses this collection is a mouse that groups together button and position data.

An **Application** collection groups together items that serve different or common purposes in a single device. An example is a keyboard that groups together the LEDs and key presses.

A **Logical** collection groups together data of various item types to form a structured data collection. An example is a relationship between the contents of a data buffer and the number of bytes the buffered data occupies. The Logical collection creates a connection between the two.

3.2 Global and Local Items

In addition to having Item types, Items can be Global or Local. Figure 5 on page 7 shows a report descriptor in which there are Items such as USAGE, USAGE PAGE, LOGICAL MINIMUM / MAXIMUM, and so on. These Items are Global and Local Items. Global Items describe the data, for example its limits, units, and bit size and count. Local Items define data parameters such as what the host is to use the data for. Following are brief descriptions of the more common Local and Global Items.

3.2.1 Global Items

Usage Page: a 32-bit value that identifies a function that the device performs, such as a control device or game controller. The upper 16 bits are the Global Usage Page Item and the lower 16 bits are the Local Usage Item. Usage items are described later in this section.

The upper 16 bits of the Global Usage Page sets the type of Usage Items available. For example, if you select Generic Desktop as the Usage Page you can include Usage Items such as a Mouse, Joystick, or Keyboard. If the Sports Controls Usage Page is selected then Usage Items such as a Golf Club or Treadmill can be selected.

Logical Maximum and Minimum: determine the limits for reported values in an array or variable. For example, a mouse that reports position values from -127 to 127 has a logical maximum of 127 and a logical minimum of -127. Another example is a single state button that is either asserted or released, and thus has a logical minimum of 0 and a logical maximum of 1.

Report Count: specifies the number of data fields in an Input, Output, or Feature Item.

Report Size: determines the size in bits of a field in an Input, Output, or Feature Item.

3.2.2 Local Items

Usage: the lower 16 bits of a 32-bit value that identifies a function that the device performs; see **Usage Page** above. The USB HID Usage Tables document, available from the USB Implementers Forum, lists and describes Usage values, or IDs, for HID devices. For example, the HID Usage Tables list the 16-bit values for mouse button, slider, keyboard button, etc.

Input, Output, and Feature items will be defined within a Usage section. For example, a Mouse will be defined with a Global Usage page of Generic Desktop, followed by a Usage item Mouse, and then a series if Input items which will report button presses and position.

Usage Minimum and Maximum: Used to assign a series of Usage IDs to the data in an array or bitmap. The Usage Minimum defines the starting ID and the Usage Maximum defines the ending ID. For example, in a 3-button mouse IDs are assigned to Button 1 (0x01), Button 2 (0x02), and Button 3 (0x03).

The Usage Minimum and Maximum entries allow a series of similar items to be defined together when the Usages are assigned to contiguous numbers. For example, instead of defining all 101 Usages separately for each key on a standard keyboard, a single Usage can be defined with the Minimum and Maximum set to the minimum and maximum key IDs. The input item for this Usage can now report any active key on the keyboard instead of just one.

3.3 Defining an Item

We will explore the definition for a Main Item as an example, which is illustrated in Figure 2. An item definition consists of a 1 byte prefix and 1 or more data description bytes. The upper 6 bits of the prefix are used to define the item, such as an Input or Collection. The lower 2 bits in the prefix indicate how many bytes will follow the prefix. The data description bytes can further classify the item and describe the type and format of the data that the item will contain.

Figure 2: Main Item Definition Template

Prefix	Data Definition	Optional Byte (If Buffered Data selected)	Reserved
8 bit	8 bit	8 bit	23 bit
nnnnnnnn	nnnnnnnx	xxxxxxx1	xxxxxxxx xxxxxxxx xxxxxxx

Input: 100000nn
Output: 100100nn
Feature: 101100nn

Input, Output, and Feature items are defined by a 2 or 3 byte string which describes the type of data being transferred. Each item has an 8-bit prefix that PSoC Creator sets up for you. An Input item has a prefix of '100000nn' where nn is the number of bytes to follow. '100100nn' represents an Output item and '101100nn' is a Feature item.

Following the prefix are up to 9 additional bits that you must configure to further define the item data. There are 23 additional bits beyond these 9 bits that are currently reserved and inaccessible. In the case of Input items used in the accompanying mouse example, there are only 8 bits (1 byte) that follow the prefix. So the Input item has a prefix of 10000001b or 0x81, which can be seen later in Figure 9, Figure 10, and Figure 11.

A second byte is required if the user selects 'Buffered Bytes' in bit 8, which causes this bit to be a value of 1. The nn of the prefix then becomes 10b and an extra byte is added to accommodate the extra data bit. Figure 3 and Figure 4 show an example of a 2-byte Input Item.

Figure 3. Two-Byte Input Item Configuration

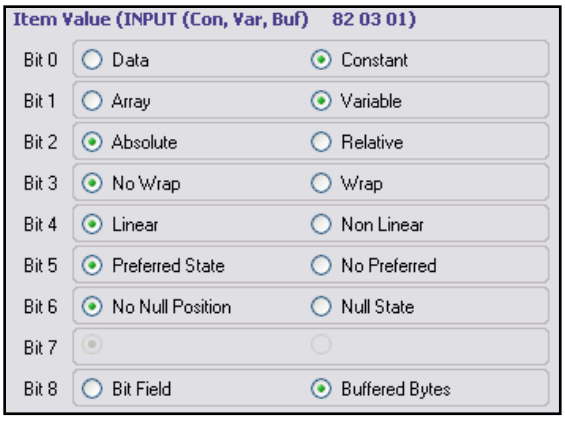
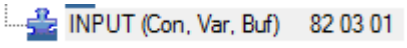


Figure 4: Two-Byte Input Item Added to HID Report Descriptor



Let's take a closer look at the example to help understand how the prefix and the bits following it are organized. Notice that Figure 3 shows an INPUT. This means we have a prefix value of '100000nn'. Notice that based on the configuration, bit 8 is set to'1', which means that there are two bytes that follow the prefix. Thus, the final prefix value is '10000010' or 0x82. Following the prefix, bit 0, bit 1, and bit 8 are set to a value of '1', giving values of 0x03 and 0x01. Put all these values together and you get 0x82, 0x03, and 0x01, which can be seen at the very top of Figure 3.

3.4 Input, Output, and Feature Bit Fields

Let us examine the 9 bits that follow the prefix byte, illustrated in Figure 3, and how each bit sets up the Input, Output and Feature items. Each bit defaults to '0' initially. For a more detailed look at these settings, refer to the section on Input, Output and Feature Items in the **USB HID Device Class Definition.**

Data versus Constant: Data means that the device data is read/write. Constant means the data is read only and cannot be modified by the host.

Array versus Variable: Array means only controls that are currently active are reported, such as a button being pressed. Variable means that the device data reported is just the current state of every control

Absolute versus Relative: Absolute means that the values are based on a fixed origin. Relative means the values represent the change to the data from the last reading. A mouse is an example of a device that provides relative data while a joystick or tablet provides absolute data.

No Wrap versus Wrap: No Wrap means a value that exceeds the set limits reports a value outside the limits. Wrap means that if the value exceeds the set maximum then it rolls over to the set minimum value. Conversely, the value rolls over to maximum if it goes below the minimum. This does not apply when using Array.

Linear versus Non-linear: Linear means the data that is measured and the value that is reported have a linear relationship with each other. This does not apply when using Array. A non-linear curve in data from a sensor is an example of when to use non-linear.

Preferred State versus No Preferred: Preferred State means that a control has a certain state to which it returns when a user is not physically interfacing with it. No Preferred means that the control remains in a certain state once a user is no longer physically interfacing with it. A joystick that returns to a center position and a push button are examples of controls with a preferred state. A toggle switch or sliders are examples of controls that have no preferred state.

No Null Position versus Null State: No Null Position means that all data sent by the control is meaningful. Null State means that the control can send meaningless data that is represented by a value outside the set logical minimum and maximum range.

Nonvolatile versus Volatile: Nonvolatile means that the device alters the value only with host interaction. Volatile means that the device has the ability to alter the value without host approval. This bit only applies to Output and Feature item report data. This does not apply when using Array.

Bit Field versus Buffered Bytes: Bit Field means that each bit in a byte can represent a specific piece of data. Buffered Bytes means that the data is represented by one or more bytes. This does not apply when using Array. A mouse or keyboard is an example of a device that uses Bit Field while a bar code reader uses Buffered Bytes.

3.5 Report Descriptor Details Summary

At a minimum, a report descriptor must contain the following Items. Additional Items provide further detail.

- Type: Input, Output, or Feature
- Usage
- Usage Page
- Logical Minimum and Maximum
- Report Size
- Report Count

At this point you have seen the basic concept of a HID device and how it is defined in a report descriptor. Let us now look at an example report descriptor for an actual HID device.

3.6 Report Descriptor for a Mouse

The easiest way to understand a HID report descriptor is to break it down and study it. One of the example projects that accompany this application note creates a simple three button mouse.

The HID report for the 3 button mouse must contain the following information:

- Button status for 3 buttons. The status will be reported as On or Off.
- X and Y axis change. The relative change in X and Y axis position is reported.

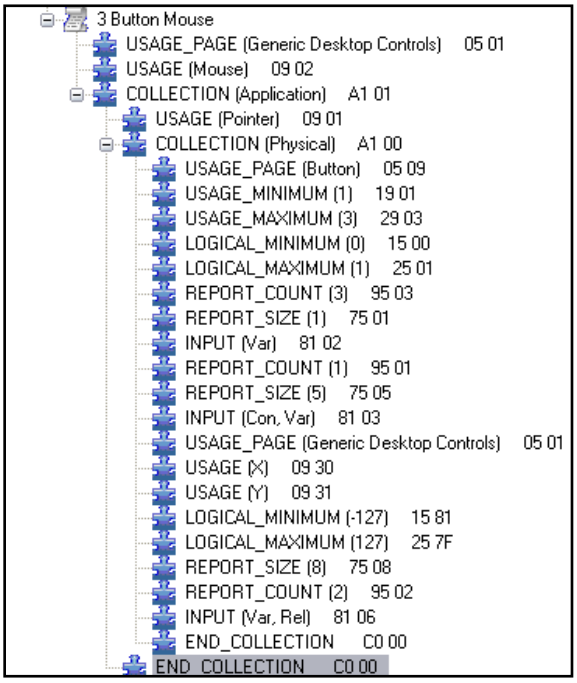
The data in the report will look like the following table:

Table 2: HID Mouse Data

Byte	0	1	2
Data	Buttons	X Axis	Y Axis
Bit 7	0	X Axis Position Change	Y Axis Position Change
Bit 6	0		
Bit 5	0		
Bit 4	0		
Bit 3	0		
Bit 2	Button 3		
Bit 1	Button 2		
Bit 0	Button 1		

Figure 5 shows the HID report descriptor for that project.

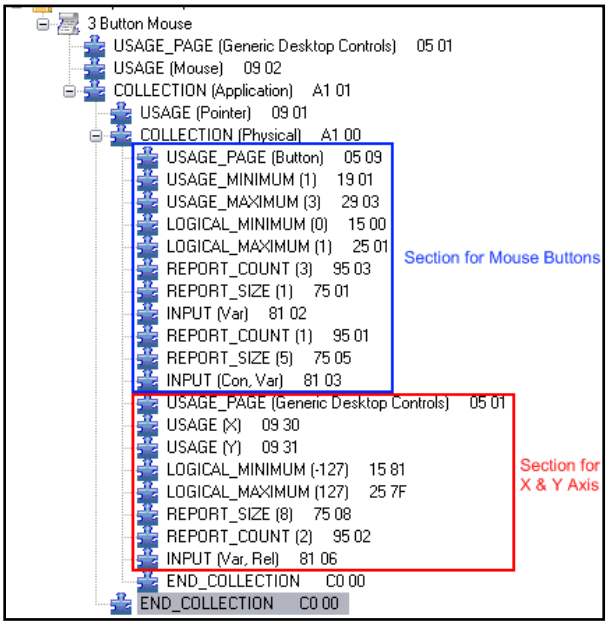
Figure 5. Mouse HID Report Descriptor



Although the report descriptor looks complicated, it is actually quite straightforward. Let us start by breaking it up into sections.

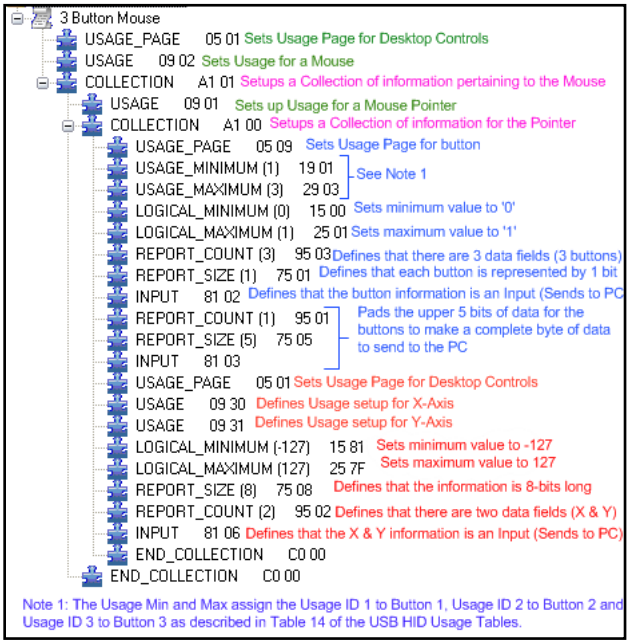
A mouse produces data based on cursor movement and buttons. The cursor has X and Y values and each button has pressed / released state information. So the report descriptor has a Collection type containing one section for the buttons and another for the cursor, as Figure 6 shows.

Figure 6. Sectioned Mouse HID Report Descriptor



The next step is to look at the function of each line in the report descriptor. By understanding how each line affects the descriptor, and knowing if the line pertains to the buttons or cursor movement, and the effect it has on the data, you will develop a greater understanding of report descriptors. Figure 7 shows a commented version of the HID mouse report descriptor detailing what each line entry does.

Figure 7. Commented Mouse HID Descriptor Report



The HID report descriptor has multiple Input Items. Each Input item is a bit field of data that tells the host about the data that is going to be sent from the mouse. Figure 8 highlights the Input Items in the report descriptor.

Figure 8. Report Descriptor Input Items

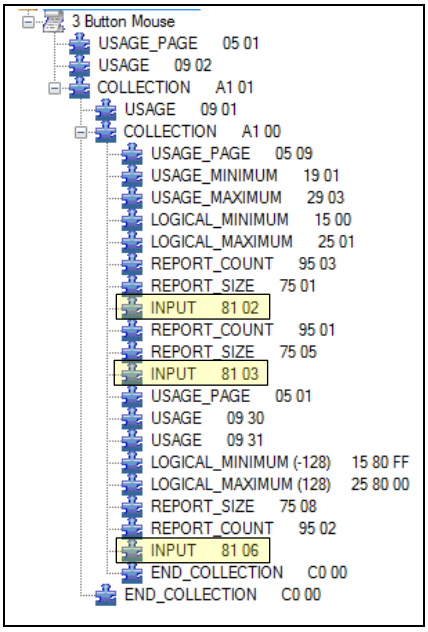


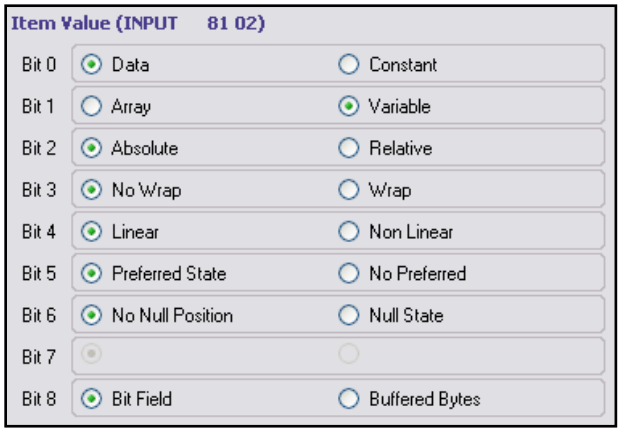
Figure 9, Figure 10, and Figure 11 show how the Input Items are configured and how the bit field is set up. This section discusses in detail what each of these settings / bits represents. In each data transaction to the host, three bytes of data will be transferred. One byte is mouse button data, another is x-axis data, and the third byte is y-axis data.

The button data is organized in a bit field format where each bit corresponds to a specific button on the mouse. Figure 9 shows the Input item setup for the three bits of button data on a three button mouse.

Each bit represents a specific button:

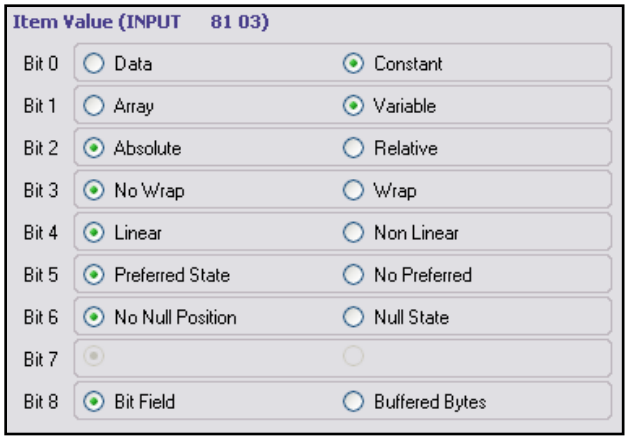
- Bit 0 = Middle Button
- Bit 1 = Right Button
- Bit 2 = Left Button

Figure 9. Input Item for Mouse Buttons



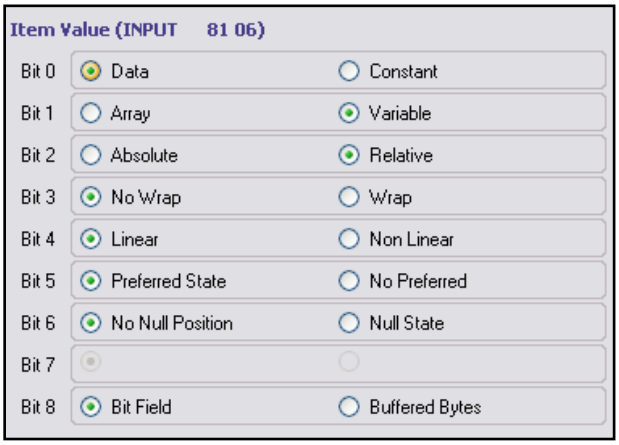
Since only three bits of a byte are used for actual data, we need to reserve the unused five bits. To do that we pad the upper five bits with zeros using a separate Input Item. Because we do not want that data to change, we use the same Input Item setup as the buttons except that we set bit 0 to Constant, as Figure 10 shows.

Figure 10. Input Item for Padding Mouse Buttons



The remaining Input Item is the X and Y information, as Figure 11 shows. Because bit 2 is set to Relative, the data shows how much X and Y have changed. The host then moves the mouse cursor accordingly.

Figure 11. Input Item for X and Y Axis



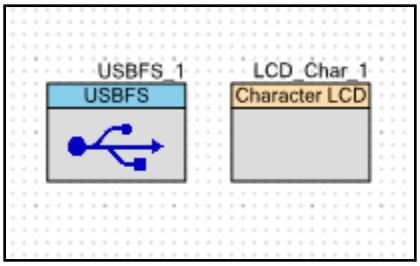
Now that you have seen the report descriptor components and the organization of a report descriptor, let us look at two projects that demonstrate how to configure a PSoC 3 or PSoC 5LP device to function as a HID mouse and a HID joystick.

4 Example Project 1: HID Mouse

The following project descriptions assume that you are familiar with developing applications using PSoC Creator for PSoC 3 or PSoC 5LP. If you are new to these products, introductions can be found in AN54181, Getting Started with PSoC 3 and AN77759, Getting Started with PSoC 5LP. If you are new to PSoC Creator, see the PSoC Creator home page.

First, open PSoC Creator and create a project named 'MyFirstHID'. Then place a USBFS Component and a Character LCD Component in the schematic entry page (*TopDesign.cysch*), as Figure 12 shows:

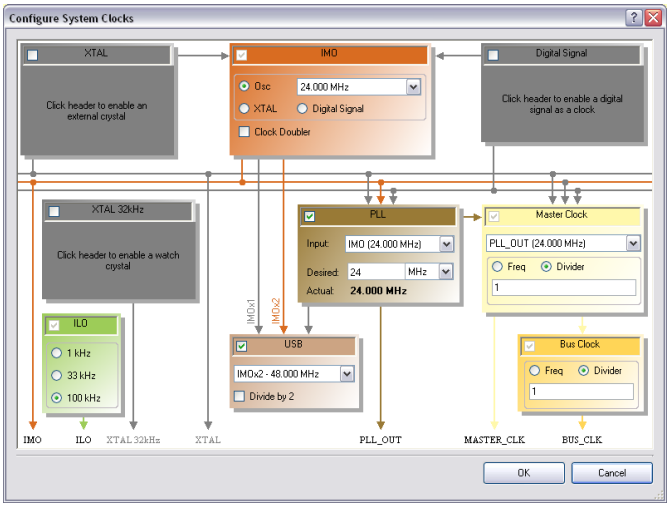
Figure 12. PSoC Creator Components for Mouse



Next, open the file *MyFirstHID.cydwr*. Click the **Clocks** tab and then double-click on one of the clocks to open the clock configuration window.

For USB operations, the USB block of the device requires a clock of 48 MHz, and the ILO must be set to 100 kHz. To get the USB clock of 48 MHz, set the IMO to 24 MHz and the USB clock to IMOx2. Adjust the PLL, USB, IMO, and ILO settings as Figure 13 shows.

Figure 13. Clocks Configuration Window



Next, select the Pins tab in *MyFirstHID.cydwr* and assign Component Pins to device pins, as Figure 14 shows. Note that the USB D+ and D- pins are automatically assigned to P15[6] and P15[7].

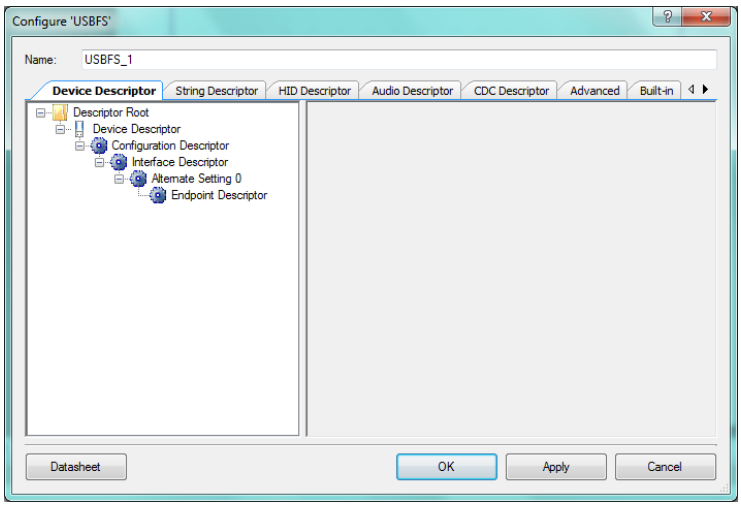
Figure 14. Device Pin Assignments

Alias	Name	Pin
dp	\USBFS_1:dp\	P15[6]
dm	\USBFS_1:dm\	P15[7]
	\LCD_Char_1:LCDPort\[6:0]	P2[6:0]

The next step is to configure the USB component, using a USB configuration wizard (see the USBFS component datasheet). The USBFS Wizard configures all of the device descriptors. When the project is built, the USB Wizard uses this information to create proper descriptor tables to interface with your PC.

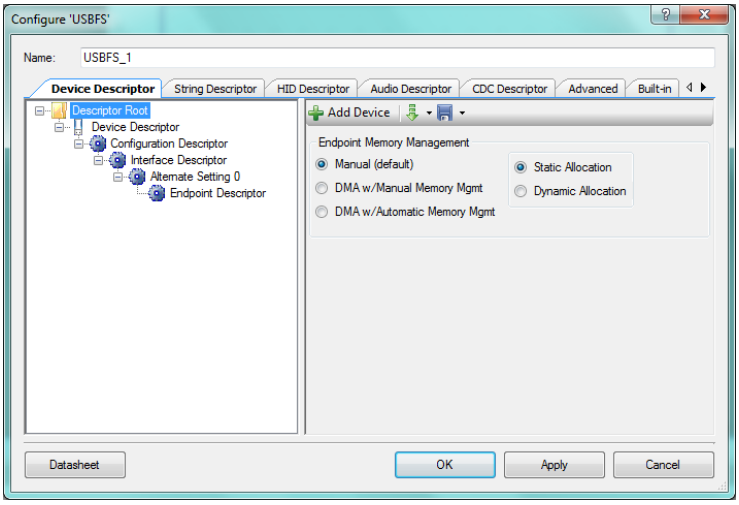
To open the configuration wizard (shown in Figure 15) double-click the USBFS Component.

Figure 15. USBFS Configuration Wizard



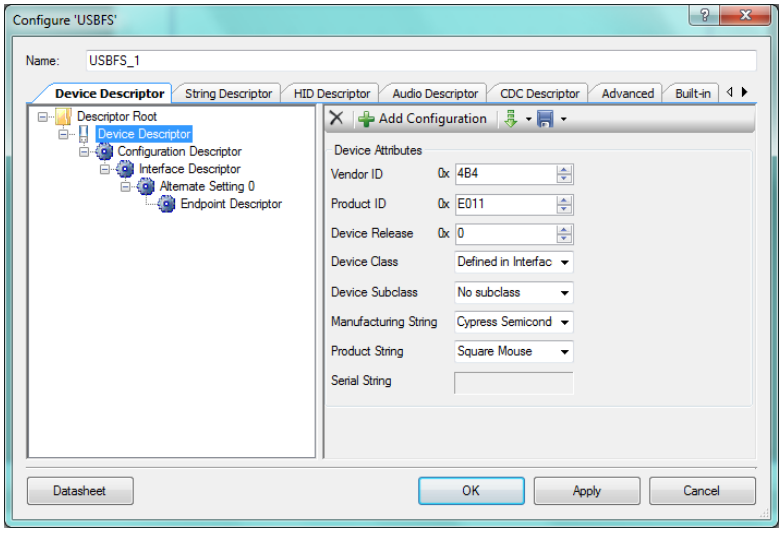
Once the wizard is open, start by clicking on **Descriptor Root** to open configuration options for Endpoint Memory Management. Ensure that the Endpoint Memory management is configured as **Manual with Static Allocation**, as Figure 16 shows. It should be configured as such by default.

Figure 16. USB Endpoint Memory Management



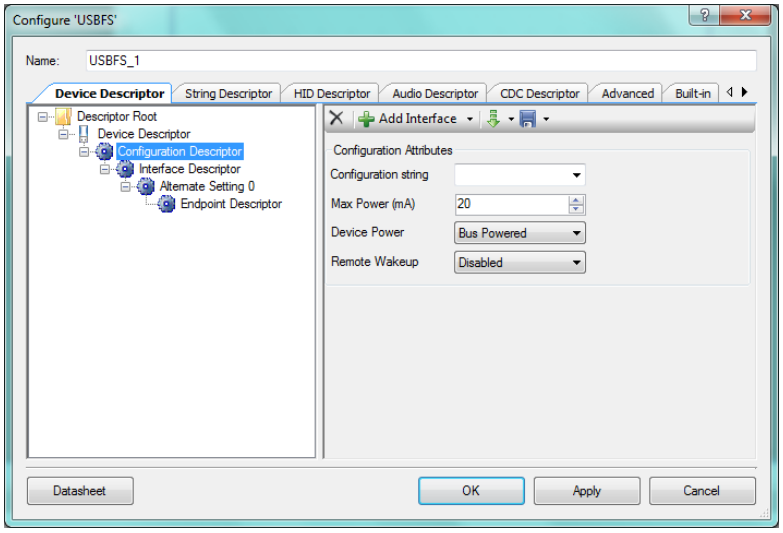
Next, click on Device Descriptor to begin its configuration. For this application, a Vendor ID (VID) and Product ID (PID) have been chosen. If you decide to create your own USB application to manufacture and sell, you must obtain a VID from the USB Implementers Forum. The PID is chosen by the designer. Adjust the device descriptor configuration so that it matches Figure 17.

Figure 17. USB Device Descriptor Setup



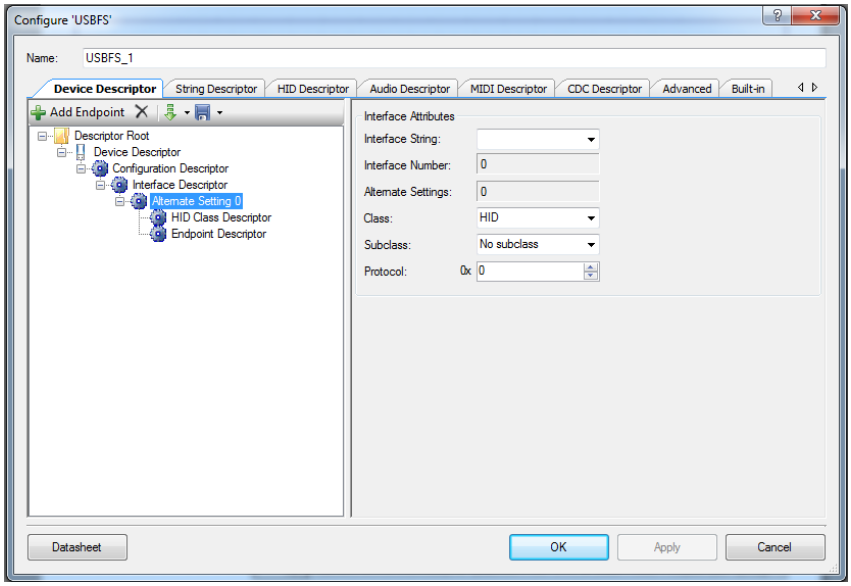
Next, select **Configuration Descriptor** and configuration options as Figure 18 shows. Because the project is bus powered, we must limit the maximum current that can be supplied to the device. It is important to specify a value that is appropriate for the device. This is due to the 100 mA / 500 mA allowance that each hub is given according to the USB 2.0 Specification. For this application, 20 mA is sufficient.

Figure 18. USB Configuration Descriptor Setup



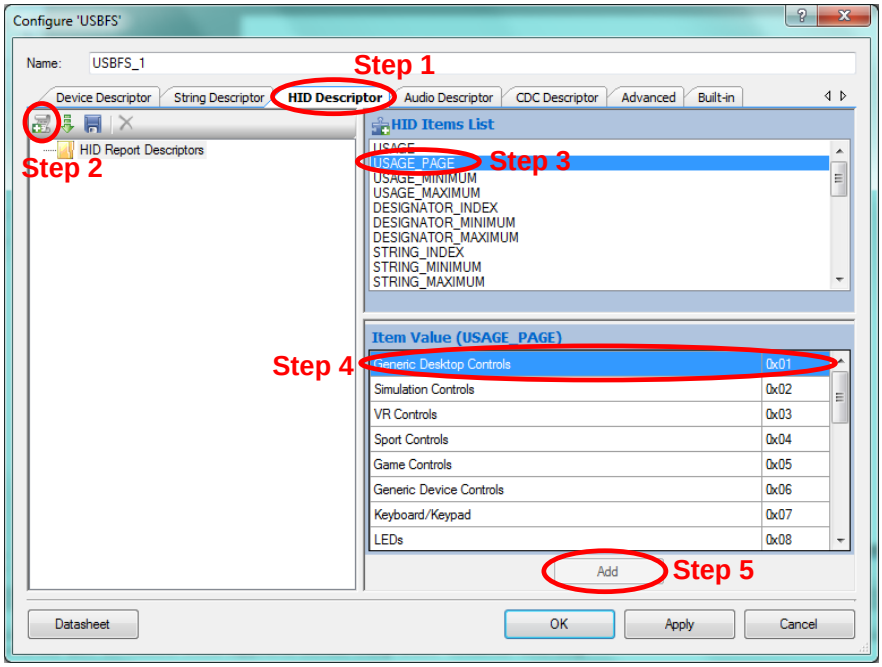
The next step is to configure the interface descriptor. To do this, click on Alternate Setting 0. In this menu, all that is required is to set the Class type to HID. When PSoC Creator generates the report descriptor, this sets the required parameters to inform the host that the attached device is a HID. Adjust the configuration options in the interface descriptor so that it matches Figure 19. Note that setting the **Class** field to be a **HID** causes a new descriptor to appear in the wizard, called a **HID Class Descriptor**.

Figure 19. USB Interface Descriptor Setup



A HID report descriptor must be associated with the interface. To do so, create a HID report descriptor - click on the HID Descriptor tab. The steps for this dialog are illustrated in Figure 20.

Figure 20. USB HID Descriptor Setup



Step 1: Click on the **HID Descriptor** tab

Step 2: Click on the **Add Report** button

Step 3: Select an Item from the **HID Item List**. Start by selecting **USAGE_PAGE**.

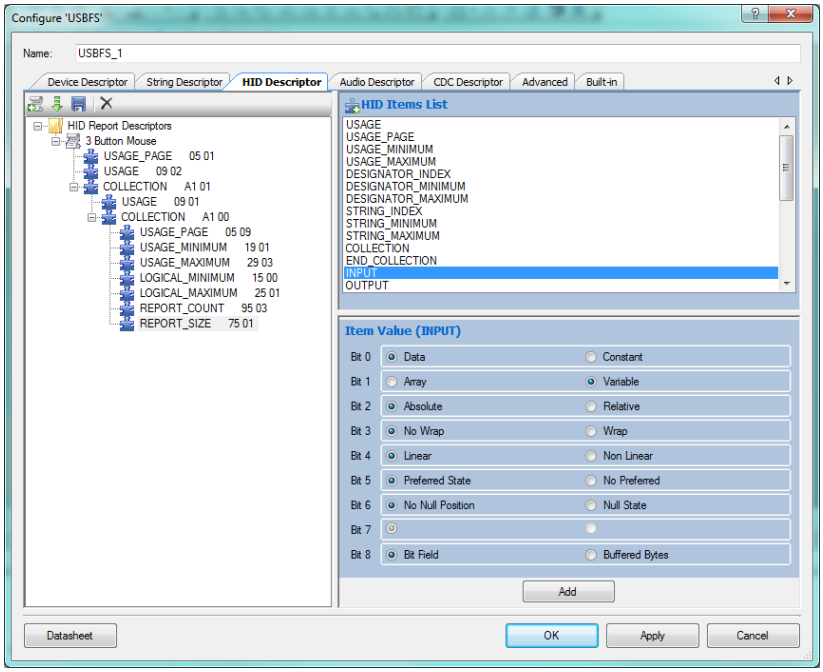
Step 4: Select a value from the **Item Value** list for the Item selected, in this case **Generic Desktop Controls**.

Step 5: Click the **Add** button to add the Item to the report descriptor.

The format for the complete HID report descriptor is shown in Figure 5 on page 7. Repeat steps 3 through 5, which are also shown in Figure 20, until the report descriptor resembles Figure 5 on page 7.

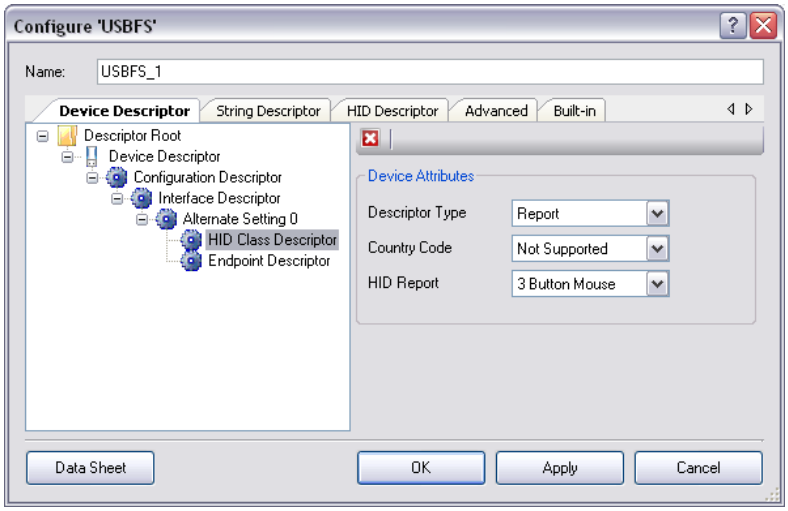
When you add and configure an Input Item, instead of the Item Value field you are presented with a window to configure the bit field, as Figure 21 shows.

Figure 21. Configuring an INPUT in a Report Descriptor



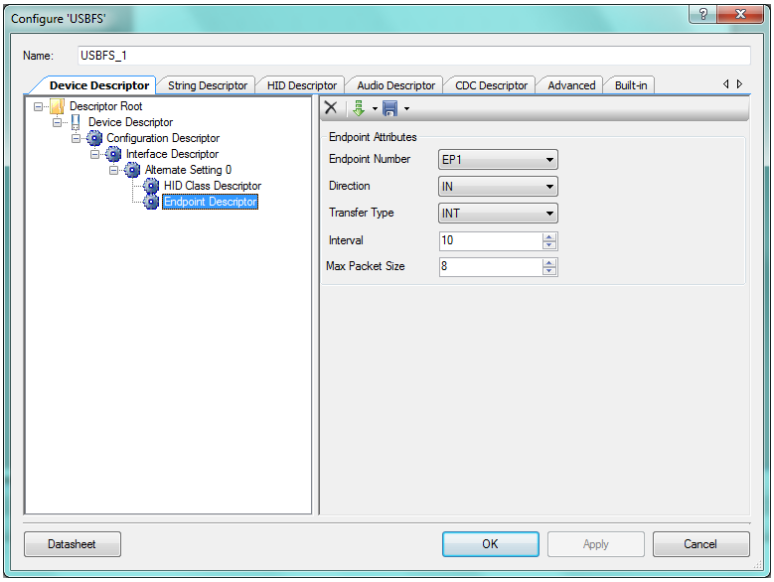
When you have completed the report descriptor, go to the HID Class Descriptor menu, as Figure 22 shows. Set the HID report to the name of the HID report that was just created. In this case, set it to **3 Button Mouse**.

Figure 22. USB Device Attributes Setup



Next, in the configuration dialog for the USBFS component, select the **Endpoint Descriptor** and, under **Endpoint Attributes**, select the direction to be **IN** and the transfer type to be **INT**, as Figure 23 shows.

Figure 23. USB Endpoint Setup



Place the following code in the file *main.c*. Note that it only takes 5 lines of code to get started with USB. The remaining code simulates mouse functionality by periodically sending data indicating that the mouse has moved. For more detailed comments, see the accompanying project file.

```
#include <device.h>

static int8 Mouse_Data[3] = {0, 0, 0};
static uint16 i = 0;

void main()
{
  CYGlobalIntEnable;
  /* Activates and configs the USB component */
  USBFS_1_Start(0, USBFS_1_3V_OPERATION);
  /* Waits for configuration data from host */
  while(!USBFS_1_bGetConfiguration());
  /* Begins initial communication with PC */
  USBFS_1_LoadInEP(1,(uint8 *)Mouse_Data,3);

  LCD_Char_1_Start();
  LCD_Char_1_Position(0,0);
  LCD_Char_1_PrintString(" My First HID ");

  for(;;)
  {
    /* Waits for ACK from the host */
    while(!USBFS_1_bGetEPAckState(1));

    /* Loads EP1 for a IN transfer to PC */
    USBFS_1_LoadInEP(1,(uint8 *)Mouse_Data, 3);

    switch (i){
    case 128:
      Mouse_Data[1] = 5;
      Mouse_Data[2] = 0;
      LCD_Char_1_Position(1, 0);
      LCD_Char_1_PrintString("Mouse Right");
      break;
```

```
        case 256:
          Mouse_Data[1] = 0;
          Mouse_Data[2] = 5;
          LCD_Char_1_Position(1, 0);
          LCD_Char_1_PrintString("Mouse Down ");
          break;

        case 384:
          Mouse_Data[1] = -5;
          Mouse_Data[2] = 0;
          LCD_Char_1_Position(1, 0);
          LCD_Char_1_PrintString("Mouse Left ");
          break;

        case 512:
          Mouse_Data[1] = 0;
          Mouse_Data[2] = -5;
          LCD_Char_1_Position(1, 0);
          LCD_Char_1_PrintString("Mouse Up   ");
          i = 0;
          break;

        default:
          break;
        }

        i++;
      }
    }
```

With all the settings properly enabled and the code entered into *main.c*, the final step is to program the PSoC 3 / PSoC 5LP on the CY8CKIT-001 Development Kit. Prior to programming the device, move the jumper on J8 from VREG to VBUS and set SW3 to 3.3 V. Because this is a HID device, we let the USB bus power the device. J8 is marked in the photo in Figure 24.

After the device is programmed, press the reset button on the DVK board, which is marked SW4, and you will see text displayed on the LCD and the cursor on your screen moving in a square pattern. Congratulations on creating your very first HID device! The generated descriptor tables for this project are located in Appendix 1.

Figure 24. CY8CKIT-001 Demonstration

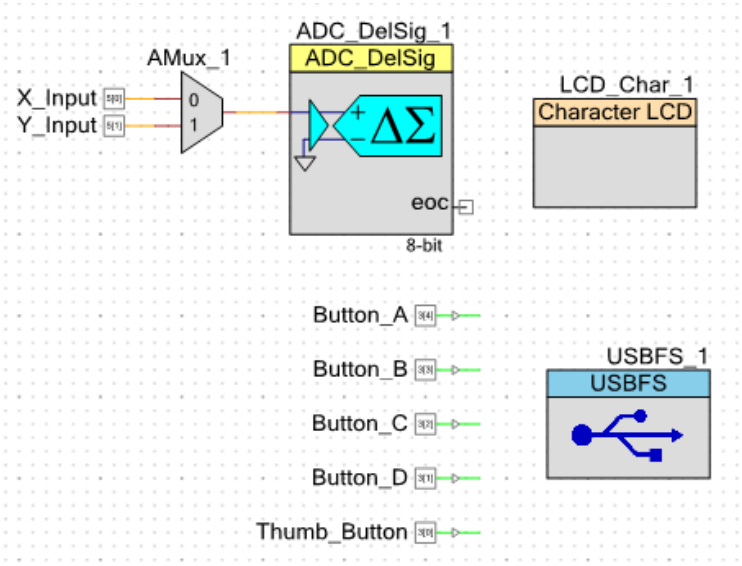
5 Example Project 2: HID Joystick

The next project is slightly more complex. For this project we built a demonstration board using a potentiometer joystick and some push buttons. To simulate a joystick using the CY8CKIT-001, use 'VR' for one axis and add another potentiometer for the other axis.

To start, create a new PSoC Creator project and name it 'HIDJoystick'. Place the following components into the schematic entry page (*TopDesign.cysch*), as Figure 25 shows:

- Delta Sigma ADC
- Analog Mux
- Character LCD
- USBFS
- (2) Analog Input Pins
- (5) Digital Input Pins

Figure 25. PSoC Creator Components for Joystick



Configure the Digital Input Pins: double-click on each Pin Component, uncheck the **HW Connection** box, and change the drive mode to **Resistive Pull Down**.

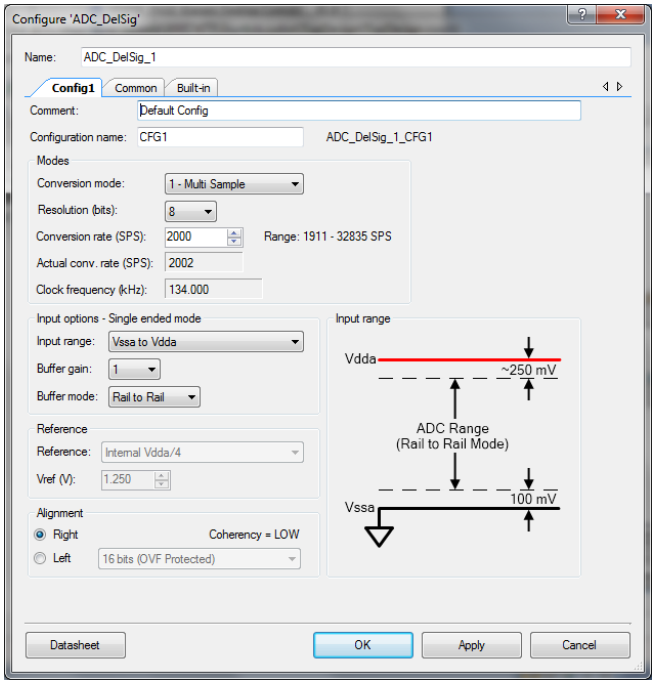
Use the same clocking configurations as in the previous project (see Figure 13 on page 11). The USB and LCD pin configurations are also the same, as Figure 26 shows. You can assign the ADC input pins and the push button pins as Figure 26 shows, or you may assign them to a GPIO pin of your choosing.

Figure 26. PSoC Creator Pin Information (Joystick)

Alias	Name	Pin
	X_Input	P5[0]
	Y_Input	P5[1]
	\LCD_Char_1:LCDPort\[6:0]	P2[6:0]
dp	\USBFS_1:dp\	P15[6]
dm	\USBFS_1:dm\	P15[7]
	Button_A	P3[4]
	Button_B	P3[3]
	Button_C	P3[2]
	Button_D	P3[1]
	Thumb_Button	P3[0]

The configuration settings for the *ADC_DelSig* are shown in Figure 27. While the conversion rate can change, it is important to keep the other parameters constant as the firmware is designed to process the ADC results based on the settings shown.

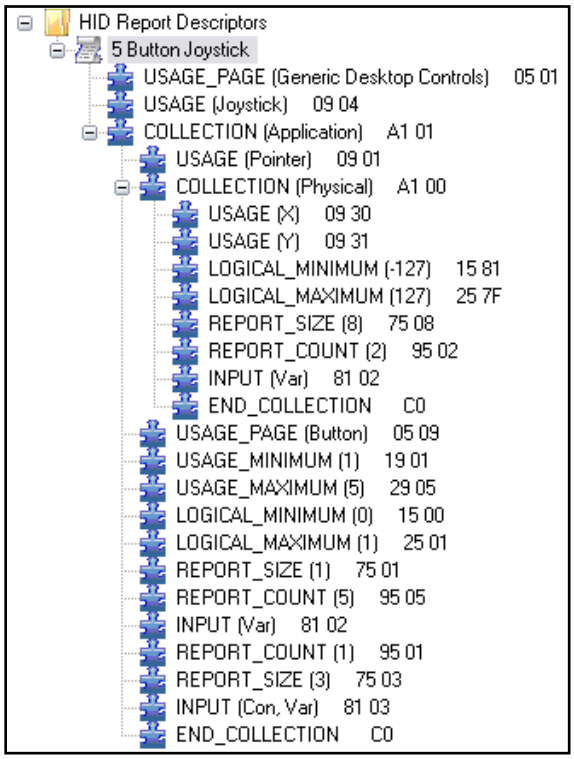
Figure 27. DelSig ADC Settings



The next step is to configure the USB settings and the HID report descriptor. Start by following the same USB configuration steps discussed in the previous project. Be sure that the PID for this project is different from that in the previous project, or the project may not function properly. This is because the host computer recognizes the previous VID/PID as a mouse. Feel free to also change the product string to a name that is joystick related. When complete, continue to create the report descriptor for the project. The report descriptor is shown in Figure 28.

The format of the descriptor layout is very similar to the mouse report descriptor with a few differences. The first difference is that the Usage for the Generic Desktop Control is set to Joystick rather than Mouse. The other noticeable difference is that unlike the report descriptor for the mouse, which specifies three buttons, this report specifies five buttons. One push button is located under the joystick and four additional buttons are located along the joystick. Similar to the mouse report descriptor, the remainder of the bits in the single byte that contains the button data is padded.

Figure 28. Joystick HID Descriptor Report



As mentioned previously, a demonstration board was built for this example project, which can be seen in Figure 29 and Figure 30. The schematic for the board is shown in Figure 31. If you do not have a potentiometer-based joystick, basic potentiometers similar to the one attached to the CY8CKIT-001 Development Kit can be used along with the push buttons on the board. Additional switches can be created by adding a button and a resistor between Vddd and GND.

Figure 29. CY8CKIT-001 Setup for Joystick Demo

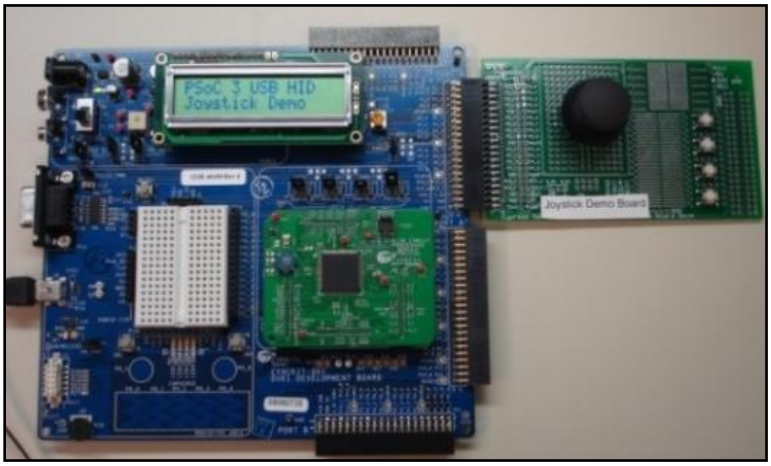


Figure 30. Close-up of Joystick Expansion Board

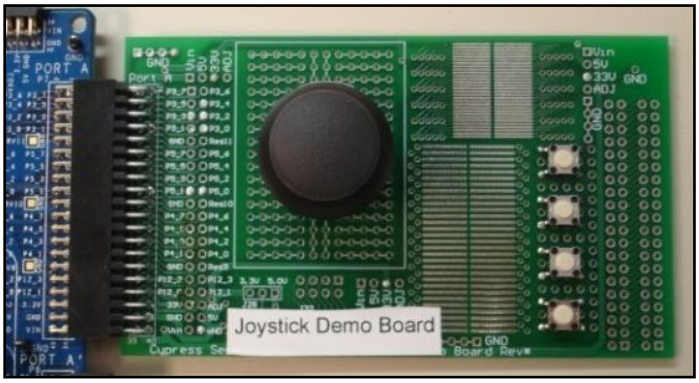


Figure 31. Schematic of Joystick Board

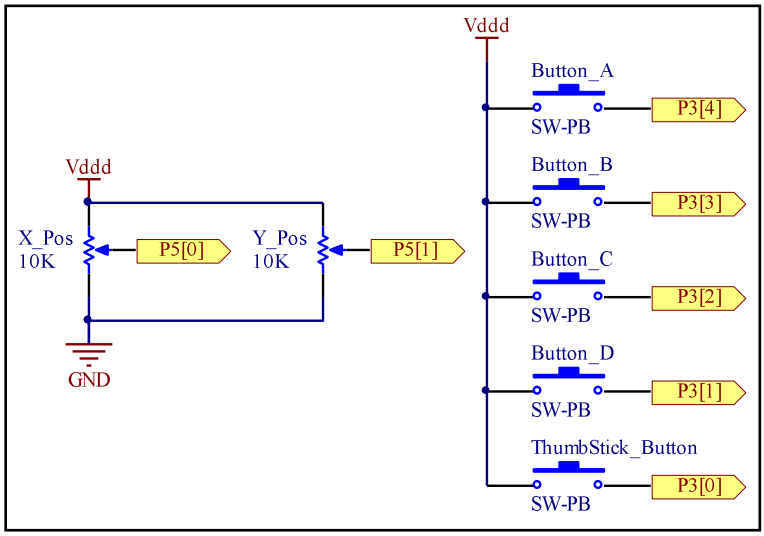


Figure 32 shows an example of a simulated joystick using the CY8CKIT-001. Five jumpers act as buttons, while the potentiometer labeled 'VR' on the development kit acts as the joystick. Note that the potentiometer drives both X and Y axis inputs, so the joystick moves in a diagonal fashion.

Figure 32. DVK Implementation of Joystick

Finally, add the following code to *main.c*. In the mouse project, 5 lines of code were identified as the core code to enable USB functionality. The same 5 lines of code are used here also. The rest of the code serves the purpose of obtaining information and preparing it to send out to the host. To see detailed comments on the code, refer to the project that accompanies this application note.

```
#include <device.h>

void StartUp (void);
void ReadJoystick (void);
void ReadButtons (void);

static uint16 X_Axis=0, Y_Axis=0;
static int16 X_Data, Y_Data;
static int8 Joystick_Data[3] = {0, 0, 0};
static unsigned char Buttons;

void main()
{
  StartUp();

  for(;;)
  {
    /* Waits for ACK from the host */
    while(!USBFS_1_bGetEPAckState(1));
    ReadJoystick();
    ReadButtons();
    /* Delay added to periodically send data */
    CyDelay(10);

    Joystick_Data[0] = X_Data;
    Joystick_Data[1] = Y_Data;
    Joystick_Data[2] = Buttons;

    /* Loads EP1 for a IN transfer to PC */
    USBFS_1_LoadInEP(
                1, (uint8 *)Joystick_Data, 3);
  }
}

void StartUp (void)
{
  CYGlobalIntEnable;

  ADC_DelSig_1_Start();
  CyIntEnable(ADC_DelSig_1_IRQ__INTC_NUMBER);
  ADC_DelSig_1_StartConvert();
  AMux_1_Start();
  AMux_1_Select(0);

  LCD_Char_1_Start();
  LCD_Char_1_Position(0,0);
  LCD_Char_1_PrintString("PSoC 3 USB HID");
  LCD_Char_1_Position(1,0);
  LCD_Char_1_PrintString("Joystick Demo");

  /* Activates and configs the USB component */
  USBFS_1_Start(0, USBFS_1_3V_OPERATION);
  /* Waits for configuration data from host */
  while(!USBFS_1_bGetConfiguration());
  /* Begins initial communication with PC */
  USBFS_1_LoadInEP(
              1, (uint8 *)Joystick_Data, 3);
}

void ReadJoystick (void)
{
  AMux_1_Select(0);
  CyDelay(1);
  ADC_DelSig_1_IsEndConversion(
```

```
                    ADC_DelSig_1_WAIT_FOR_RESULT);
  X_Axis = ADC_DelSig_1_GetResult16();

  AMux_1_Select(1);
  CyDelay(1);
  ADC_DelSig_1_IsEndConversion(
                    ADC_DelSig_1_WAIT_FOR_RESULT);
  Y_Axis = ADC_DelSig_1_GetResult16();

  X_Data = X_Axis - 127;
  Y_Data = Y_Axis - 127;

  if(X_Data > 127)
    X_Data = 127;
  if(Y_Data > 127)
    Y_Data = 127;
  if(X_Data < -127)
    X_Data = -127;
  if(Y_Data< -127)
    Y_Data = -127;

  Y_Data = Y_Data * -1;
}

void ReadButtons (void)
{
  if(Thumb_Button_Read() != 0)
    Buttons |= 0x01;
  else
    Buttons &= ~0x01;

  if(Button_A_Read() != 0)
    Buttons |= 0x02;
  else
    Buttons &= ~0x02;

  if(Button_B_Read() != 0)
    Buttons |= 0x04;
  else
    Buttons &= ~0x04;

  if(Button_C_Read() != 0)
    Buttons |= 0x08;
  else
    Buttons &= ~0x08;

  if(Button_D_Read() != 0)
    Buttons |= 0x10;
  else
    Buttons &= ~0x10;
}
```

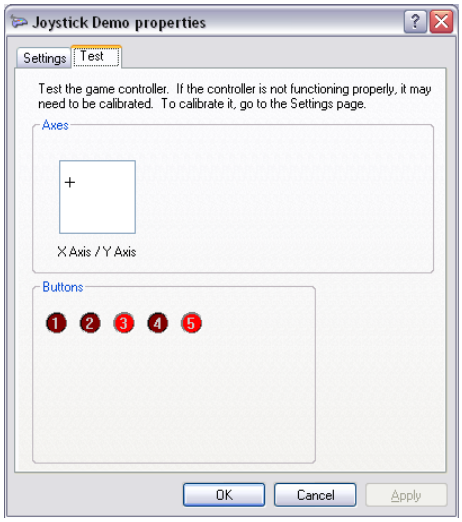
With all of the settings properly enabled and the code placed into *main.c*, the final step is to program the PSoC 3 or PSoC 5LP on the CY8CKIT-001 Development Kit. Similar to Project 1, ensure that prior to programming the device, the jumper on J8 is moved from VREG to VBUS and SW3 is set to 5.0 V. After the device is programmed, press the reset button on the board, which is labeled as SW4.

How to test and verify the joystick depends on the PC operating system that you are using:

5.1 Testing in Windows XP

On your PC, open Windows Control Panel > Game Controllers. Then select the attached joystick, which should be labeled "Joystick Demo". Click on "Properties" and then click on the "Test Tab". Move the joystick around and press buttons. The result you see on the screen is shown in Figure 33.

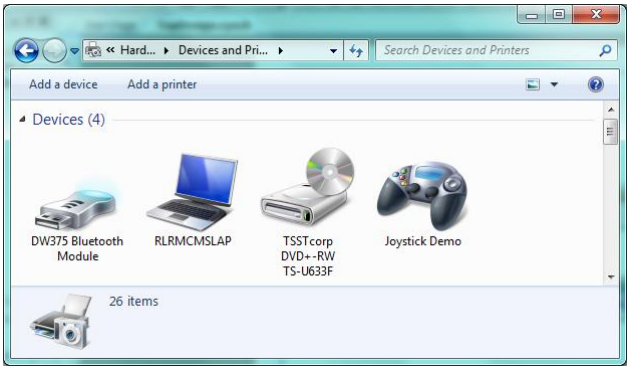
Figure 33. Joystick Test in Windows Control Panel



5.2 Testing in Windows Vista and Windows 7

Open the Windows Control Panel and navigate to Hardware and Sound > Devices and Printers and locate Joystick Demo under the Devices category as shown in Figure 34.

Figure 34. Locating the Joystick in Devices and Printer



Right click on "Joystick Demo" and select "Game Controller Settings". Click on "Properties" and then click on the "Test Tab". Proceed to test the joystick as was shown in Figure 33 on page 24.

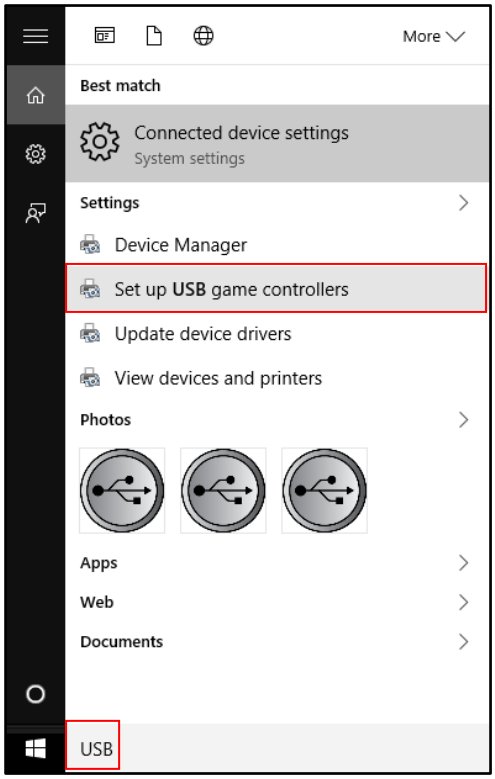
Another way to test the joystick is to simply open and play your favorite video game.

The generated descriptor tables for this project are located in Appendix 2.

5.3 Testing in Windows 10

Search for “USB” in the Windows 10 start menu. Within the results, you will see the “Set up USB game controllers” option under the “Settings” heading. Click this action to start the configuration of the joystick.

Figure 35. Configuring Joystick in Windows 10



After the configuration window appears, select “Joystick Demo” and click the “Properties” button. Click the “Test” tab and test the joystick, as was shown in Figure 33 on page 24.

6 Summary

This application note has shown how to create an input HID device. You can apply the principles presented here to a wide variety of HID input devices.

To add more HID capability to your PSoC 3 or PSoC 5LP project, you can support output transactions and composite devices – see the more advanced AN58726.

You can go beyond implementing HID devices and transfer any kind of data between PSoC and your PC. To use the HID interface to do this, see AN82072. To use any of the USB transfer types for the same purpose, see AN56377.

7 References

For more information regarding HID and to learn just how deeply you can dive into HID, refer to the Device Class Definition for Human Interface Devices Firmware Specification, which can be acquired from www.usb.org. A good reference is the HID Usage Tables, which can also be acquired from www.usb.org. Another great reference and learning tool for both HID and USB is a book available for purchase titled *USB Complete*, by Jan Axelson.

8 Additional Resources

Application Notes:

- AN57294 - USB 101: An Introduction to Universal Serial Bus 2.0
- AN58726 - USB HID Intermediate with PSoC® 3 and PSoC 5LP
- AN82072 - PSoC® 3 and PSoC 5LP USB General Data Transfer with Standard HID Drivers
- AN56377 - PSoC® 3 / PSoC 5LP USB Transfer Types

Related Resources:

- USB HID Device Class Definition

9 About the Author

Name:	Robert Murphy
Title:	Sr. Staff Systems Engineer
Background:	Robert Murphy graduated from Purdue University with a Bachelor's Degree in Electrical Engineering Technology.

A Descriptor Tables for Project 1

```
/*******************************************************************
 Device Descriptors
 *******************************************************************
/* Descriptor Length                    */ 0x12u,
/* DescriptorType: DEVICE               */ 0x01u,
/* bcdUSB (ver 2.0)                     */ 0x00u, 0x02u,
/* bDeviceClass                         */ 0x00u,
/* bDeviceSubClass                      */ 0x00u,
/* bDeviceProtocol                      */ 0x00u,
/* bMaxPacketSize0                      */ 0x08u,
/* idVendor                             */ 0xB4u, 0x04u,
/* idProduct                            */ 0x00u, 0x10u,
/* bcdDevice                            */ 0x00u, 0x00u,
/* iManufacturer                        */ 0x01u,
/* iProduct                             */ 0x02u,
/* iSerialNumber                        */ 0x00u,
/* bNumConfigurations                   */ 0x01u

/*******************************************************************
 Config Descriptor
 *******************************************************************/
/*  Config Descriptor Length            */ 0x09u,
/*  DescriptorType: CONFIG              */ 0x02u,
/*  wTotalLength                        */ 0x22u, 0x00u,
/*  bNumInterfaces                      */ 0x01u,
/*  bConfigurationValue                 */ 0x01u,
/*  iConfiguration                      */ 0x00u,
/*  bmAttributes                        */ 0x80u,
/*  bMaxPower                           */ 0x0Au,

/*******************************************************************
 Interface Descriptor
 *******************************************************************/
/*  Interface Descriptor Length         */ 0x09u,
/*  DescriptorType: INTERFACE           */ 0x04u,
/*  bInterfaceNumber                    */ 0x00u,
/*  bAlternateSetting                   */ 0x00u,
/*  bNumEndpoints                       */ 0x01u,
/*  bInterfaceClass                     */ 0x03u,
/*  bInterfaceSubClass                  */ 0x00u,
/*  bInterfaceProtocol                  */ 0x00u,
/*  iInterface                          */ 0x00u,

/*******************************************************************
 HID Class Descriptor
 *******************************************************************/
/*  HID Class Descriptor Length         */ 0x09u,
/*  DescriptorType: HID_CLASS           */ 0x21u,
/*  bcdHID                              */ 0x11u, 0x01u,
/*  bCountryCode                        */ 0x00u,
/*  bNumDescriptors                     */ 0x01u,
/*  bDescriptorType                     */ 0x22u,
/*  wDescriptorLength (LSB)             */ USBFS_1_HID_RPT_1_SIZE_LSB,
/*  wDescriptorLength (MSB)             */ USBFS_1_HID_RPT_1_SIZE_MSB,

/*******************************************************************
 Endpoint Descriptor
 *******************************************************************/
/*  Endpoint Descriptor Length          */ 0x07u,
/*  DescriptorType: ENDPOINT            */ 0x05u,
/*  bEndpointAddress                    */ 0x81u,
/*  bmAttributes                        */ 0x03u,
/*  wMaxPacketSize                      */ 0x08u, 0x00u,
/*  bInterval                           */ 0x0Au

/*******************************************************************
```

```
 String Descriptor Table
 ***************************************************************************/

/***************************************************************************
 Language ID Descriptor
 ***************************************************************************/
/* Descriptor Length                        */ 0x04u,
/* DescriptorType: STRING                   */ 0x03u,
/* Language Id                              */ 0x09u, 0x04u,

/***************************************************************************
 String Descriptor: "Cypress Semiconductor"
 ***************************************************************************/
/* Descriptor Length                        */ 0x2Cu,
/* DescriptorType: STRING                   */ 0x03u,
 'C', 0,'y', 0,'p', 0,'r', 0,'e', 0,'s', 0,'s', 0,' ', 0,'S', 0,'e', 0
,'m', 0,'i', 0,'c', 0,'o', 0,'n', 0,'d', 0,'u', 0,'c', 0,'t', 0,'o', 0
,'r', 0,

/***************************************************************************
 String Descriptor: "Square Mouse"
 ***************************************************************************/
/* Descriptor Length                        */ 0x1Au,
/* DescriptorType: STRING                   */ 0x03u,
 'S', 0,'q', 0,'u', 0,'a', 0,'r', 0,'e', 0,' ', 0,'M', 0,'o', 0,'u', 0
,'s', 0,'e', 0,

/***************************************************************************
 Serial Number String Descriptor
/* Descriptor Length                        */ 0x02u,
/* DescriptorType: STRING                   */ 0x03u

/***************************************************************************
 HID Report Descriptor: 3 Button Mouse
 ***************************************************************************/
/* Descriptor Size (Not part of descriptor)*/ 0x32u, 0x00u,
/* USAGE_PAGE                               */ 0x05u, 0x01u,
/* USAGE                                    */ 0x09u, 0x02u,
/* COLLECTION                               */ 0xA1u, 0x01u,
/* USAGE                                    */ 0x09u, 0x01u,
/* COLLECTION                               */ 0xA1u, 0x00u,
/* USAGE_PAGE                               */ 0x05u, 0x09u,
/* USAGE_MINIMUM                            */ 0x19u, 0x01u,
/* USAGE_MAXIMUM                            */ 0x29u, 0x03u,
/* LOGICAL_MINIMUM                          */ 0x15u, 0x00u,
/* LOGICAL_MAXIMUM                          */ 0x25u, 0x01u,
/* REPORT_COUNT                             */ 0x95u, 0x03u,
/* REPORT_SIZE                              */ 0x75u, 0x01u,
/* INPUT                                    */ 0x81u, 0x02u,
/* REPORT_COUNT                             */ 0x95u, 0x01u,
/* REPORT_SIZE                              */ 0x75u, 0x05u,
/* INPUT                                    */ 0x81u, 0x03u,
/* USAGE_PAGE                               */ 0x05u, 0x01u,
/* USAGE                                    */ 0x09u, 0x30u,
/* USAGE                                    */ 0x09u, 0x31u,
/* LOGICAL_MINIMUM                          */ 0x15u, 0x80u,
/* LOGICAL_MAXIMUM                          */ 0x25u, 0x7Fu,
/* REPORT_SIZE                              */ 0x75u, 0x08u,
/* REPORT_COUNT                             */ 0x95u, 0x02u,
/* INPUT                                    */ 0x81u, 0x06u,
/* END_COLLECTION                           */ 0xC0u,
/* END_COLLECTION                           */ 0xC0u,
```

B Descriptor Tables for Project 2

```
/*******************************************************************
 Device Descriptors
 *******************************************************************/
/* Descriptor Length                    */ 0x12u,
/* DescriptorType: DEVICE               */ 0x01u,
/* bcdUSB (ver 2.0)                     */ 0x00u, 0x02u,
/* bDeviceClass                         */ 0x00u,
/* bDeviceSubClass                      */ 0x00u,
/* bDeviceProtocol                      */ 0x00u,
/* bMaxPacketSize0                      */ 0x08u,
/* idVendor                             */ 0xB4u, 0x04u,
/* idProduct                            */ 0x0Fu, 0x21u,
/* bcdDevice                            */ 0x00u, 0x00u,
/* iManufacturer                        */ 0x02u,
/* iProduct                             */ 0x03u,
/* iSerialNumber                        */ 0x00u,
/* bNumConfigurations                   */ 0x01u

/*******************************************************************
 Config Descriptor
 *******************************************************************/
/*  Config Descriptor Length            */ 0x09u,
/*  DescriptorType: CONFIG              */ 0x02u,
/*  wTotalLength                        */ 0x22u, 0x00u,
/*  bNumInterfaces                      */ 0x01u,
/*  bConfigurationValue                 */ 0x01u,
/*  iConfiguration                      */ 0x00u,
/*  bmAttributes                        */ 0x80u,
/*  bMaxPower                           */ 0x00u,

/*******************************************************************
 Interface Descriptor
 *******************************************************************/
/*  Interface Descriptor Length         */ 0x09u,
/*  DescriptorType: INTERFACE           */ 0x04u,
/*  bInterfaceNumber                    */ 0x00u,
/*  bAlternateSetting                   */ 0x00u,
/*  bNumEndpoints                       */ 0x01u,
/*  bInterfaceClass                     */ 0x03u,
/*  bInterfaceSubClass                  */ 0x00u,
/*  bInterfaceProtocol                  */ 0x00u,
/*  iInterface                          */ 0x00u,

/*******************************************************************
 HID Class Descriptor
 *******************************************************************/
/*  HID Class Descriptor Length         */ 0x09u,
/*  DescriptorType: HID_CLASS           */ 0x21u,
/*  bcdHID                              */ 0x11u, 0x01u,
/*  bCountryCode                        */ 0x00u,
/*  bNumDescriptors                     */ 0x01u,
/*  bDescriptorType                     */ 0x22u,
/*  wDescriptorLength (LSB)             */ USBFS_1_HID_RPT_1_SIZE_LSB,
/*  wDescriptorLength (MSB)             */ USBFS_1_HID_RPT_1_SIZE_MSB,

/*******************************************************************
 Endpoint Descriptor
 *******************************************************************/
/*  Endpoint Descriptor Length          */ 0x07u,
/*  DescriptorType: ENDPOINT            */ 0x05u,
/*  bEndpointAddress                    */ 0x81u,
/*  bmAttributes                        */ 0x03u,
/*  wMaxPacketSize                      */ 0x08u, 0x00u,
/*  bInterval                           */ 0x0Au
/*******************************************************************
 String Descriptor Table
 *******************************************************************/
```

```
/*******************************************************************
 Language ID Descriptor
 *******************************************************************/
/* Descriptor Length                        */ 0x04u,
/* DescriptorType: STRING                   */ 0x03u,
/* Language Id                              */ 0x09u, 0x04u,

/*******************************************************************
 String Descriptor: "Cypress Semiconductor"
 *******************************************************************/
 /* Descriptor Length                        */ 0x2Cu,
 /* DescriptorType: STRING                   */ 0x03u,
 'C', 0,'y', 0,'p', 0,'r', 0,'e', 0,'s', 0,'s', 0,' ', 0,'S', 0,'e', 0
,'m', 0,'i', 0,'c', 0,'o', 0,'n', 0,'d', 0,'u', 0,'c', 0,'t', 0,'o', 0
,'r', 0,

/*******************************************************************
 String Descriptor: "Joystick Demo"
 *******************************************************************/
 /* Descriptor Length                        */ 0x1Cu,
 /* DescriptorType: STRING                   */ 0x03u,
 'J', 0,'o', 0,'y', 0,'s', 0,'t', 0,'i', 0,'c', 0,'k', 0,' ', 0,'D', 0
,'e', 0,'m', 0,'o', 0,

/*******************************************************************
 Serial Number String Descriptor
 *******************************************************************/
/* Descriptor Length                        */ 0x02u,
/* DescriptorType: STRING                   */ 0x03u

/*******************************************************************
 HID Report Descriptor: HID Report Descriptor 1
 *******************************************************************/
/* Descriptor Size (Not part of descriptor)*/ 0x30u, 0x00u,
/* USAGE_PAGE                               */ 0x05u, 0x01u,
/* USAGE                                    */ 0x09u, 0x04u,
/* COLLECTION                               */ 0xA1u, 0x01u,
/* USAGE                                    */ 0x09u, 0x01u,
/* COLLECTION                               */ 0xA1u, 0x00u,
/* USAGE                                    */ 0x09u, 0x30u,
/* USAGE                                    */ 0x09u, 0x31u,
/* LOGICAL_MINIMUM                          */ 0x15u, 0x81u,
/* LOGICAL_MAXIMUM                          */ 0x25u, 0x7Fu,
/* REPORT_SIZE                              */ 0x75u, 0x08u,
/* REPORT_COUNT                             */ 0x95u, 0x02u,
/* INPUT                                    */ 0x81u, 0x02u,
/* END_COLLECTION                           */ 0xC0u,
/* USAGE_PAGE                               */ 0x05u, 0x09u,
/* USAGE_MINIMUM                            */ 0x19u, 0x01u,
/* USAGE_MAXIMUM                            */ 0x29u, 0x05u,
/* LOGICAL_MINIMUM                          */ 0x15u, 0x00u,
/* LOGICAL_MAXIMUM                          */ 0x25u, 0x01u,
/* REPORT_SIZE                              */ 0x75u, 0x01u,
/* REPORT_COUNT                             */ 0x95u, 0x05u,
/* INPUT                                    */ 0x81u, 0x02u,
/* REPORT_COUNT                             */ 0x95u, 0x01u,
/* REPORT_SIZE                              */ 0x75u, 0x03u,
/* INPUT                                    */ 0x81u, 0x03u,
/* END_COLLECTION                           */ 0xC0u,
```

Document History

Document Title: AN57473 - USB HID Basics with PSoC® 3 and PSoC 5LP

Document Number: 001-57473

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2806313	RLRM	11/12/2009	New application note.
*A	2990432	RLRM	08/25/2010	Updated images.
*B	3087682	RLRM	11/17/2010	Added Report Descriptor section.
*C	3221647	RLRM	04/10/2011	Updated title, abstract, and figures. Performed copyedit.
*D	3453501	RLRM	12/02/2011	Updated template according to current Cypress standards.
*E	3488248	RLRM	01/27/2012	Updated the following sections: Introduction Introduction to HID Report Descriptor for a Mouse Added Testing in Windows Vista and Windows 7 section
*F	3809617	DASG	11/12/2012	Updated for PSoC 5LP
*G	4028919	MKEA	06/15/2013	Changed Title, Abstract, Introduction, updated keywords in document properties, updated related application notes section. Miscellaneous formatting and readability improvements.
*H	4209154	RLRM	12/03/2013	Updated projects to Creator 3.0 and updated relevant figures. Added information about testing joystick project in Windows 8.
*I	4514219	KLMZ	11/05/2014	Generalized the item definition section, added a graphic to show the main item definition template Added link to USB HID class specification to related resources, referenced this link in the bit fields section Added clarifying content to Usage Page, Usage, and Usage Min/Max sections Added description and table of Mouse report data before showing the descriptor
*J	5219734	RLRM	07/13/2016	Updated project files to PSoC Creator 3.3. Updated template
*K	5687768	AESATMP7	04/18/2017	Updated Cypress Logo and Copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709