

USB Gamepad

By: Lynn Nguyen
Associated Project: Yes
Associated Part Family: CY8C24894
PSoC Designer Version: 4.4
Referenced Application Notes: None

Summary

This application note explains how to implement a USB gamepad with a joystick and 2 buttons using the CY3214 PSoCEval USB board. We explain how a developer can create their Human Interface Device (HID) reports for USB communications.

Introduction

The PSoC controller board is often connected to a ‘complete’ machine such as a PC for programming or debugging reasons. Sometimes a ‘host machine’ is needed to configure and control the operations of the PSoC functions. In this role, the PC acts as a ‘host controller.’ One way to connect this particular PSoC device to the PC is through USB. USB stands for Universal Serial Bus and it is a standard to interface devices. The USB standard defines physical, electrical, and interface protocol characteristics for connecting devices. USB is a newer technology and is intended to retire other forms of communication such as conventional parallel and serial ports on PCs. The standard specifies different types of devices that can be connected over this interface, such as peripherals, display, audio/video. One such type of a device is referred to as an HID or human interface device. This is exactly what it sounds like. It is a device that humans use to interact with something—for example, a mouse. In this application note, we use a gamepad, implemented on the microcontroller board, as an HID to connect to the host PC.

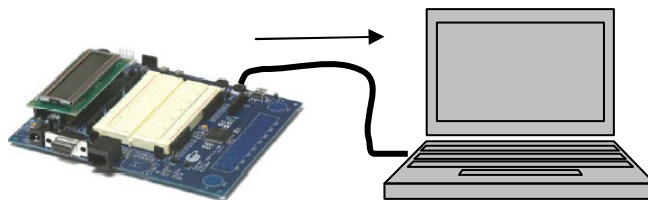


Figure 1 – Gamepad and Host Machine

Before we describe how this HID is connected to the PC, we take a brief detour into how real-life analog signal input is converted into a digital input that the host controller can process. An analog signal represents a waveform, say, of voltage on a line over time. Such signals can be supplied by the microcontroller board by connecting them to analog input puts. In this case, two analog signals, X and Y from the joystick on the microcontroller board are connected as analog inputs to the GPIO (General Purpose Input-Output) pines.

The pins are then connected to an analog mux bus which is in turn connected to a PGA. The output of the PGA is sent to a 10-bit ADC. The processor reads from the ADC to measure the voltages on the X or Y axes. The 2 buttons that we will be using are connected to pins using pull-down resistors. Once the processor has all the information from the axes and the buttons, it sends the information to the PC using an HID report through the USB interface.

- **ADC** – Component that takes an analog signal as input and produces a digital output. A 10-bit ADC will produce a 10-bit value representing the value of the analog signal at a given point in time. An ADC works by sampling an input signal at different time points. The sampled value is then quantized into a number, say, 1.2 voltage. The quantized value is represented in a binary number by appropriately scaling the sampled and quantized value: say 0000 1100 representing a binary representation of 12.
- **PGA** – The PGA is used to connect the ADC to the pin. The main purpose is to scale the maximum signal it reads from the pin to the maximum input of the ADC.
- **Mux** – Selects one of many analog or digital input signals and outputs into a single line. This makes it possible for several signals to share a device/resource (such as a PGA) instead of having one device per input signal.

Creating the Project

Create a new project in PSoC Designer for part CY8C24894-24LFXI and have the main file be generated with C.

Module Selection

For this project, we will want to choose:

1. ADCINC (Select Single Stage Modulator) found under *ADCs*
2. PGA founder under *Amplifiers*
3. USBFS (Select Human Interface Device) found under *Protocols*.



Figure 2 – Selected Modules

Also, for simplicity, I've renamed the modules.

Module Configuration

Now, go to the *Interconnect View*. Here we will configure the modules we've selected.

1. Right click ADC and select place. In the middle window on the left, set the following:

User Module Parameters	Value
DataFormat	Unsigned
Resolution	10 Bit
Data Clock	VC1
ClockPhase	Normal
PosInput	ACB00
NegInput	ACB00
NegInputGain	Disconnected
PulseWidth	1
PWM Output	None

Figure 3 – ADC configuration

2. Right click PGA and select place. Now modify the configurations on the left window to make it look like the following image:

User Module Parameters	Value
Gain	1.000
Input	AnalogColumnMUXBusSwitch_0
Reference	VSS
AnalogBus	AnalogOutBus_0

Figure 4 – PGA Configuration

3. Now comes the fun part... we must define a HID class for this device so that it will enumerate correctly as a gamepad when plugged into the PC. Right click the USBFS module place it. Then, right click again and select *USB Setup Wizard...*
4. We'll first define the HID report. To do this, we'll set the data in the bottom *Descriptor* window. First, click on *Add HID Report* on the right hand side and name this report GAMEPAD_RPT. Then, continuously click on *Insert HID Item* and populate the window to look like Figure 4.

Descriptor	Data
HID Report Descriptor Root	USBFS
HID Report Descriptor	GAMEPAD_RPT
Usage Page	Usage Page 05 01
Usage	Usage 09 05
Collection	Collection (Application 01)
Usage	Usage 09 01
Collection	Collection (Physical 00)
Usage	Usage 09 30
Usage	Usage 09 31
Logical Minimum	Logical Minimum 15 00
Logical Maximum	Logical Maximum 26 FF 00
Physical Minimum	Physical Minimum 35 00
Physical Maximum	Physical Maximum 46 FF 03
Report Size	Report Size 75 10
Report Count	Report Count 95 02
Input	Input (Data, Variable, Absolute 02)
End Collection	End Collection C0
Usage Page	Usage Page 05 09
Usage Minimum	Usage Minimum 19 01
Usage Maximum	Usage Maximum 29 02
Logical Minimum	Logical Minimum 15 00
Logical Maximum	Logical Maximum 25 01
Report Size	Report Size 75 01
Report Count	Report Count 95 02
Input	Input (Data, Variable, Absolute 02)
Report Size	Report Size 75 01
Report Count	Report Count 95 06
Input	Input (Constant 03)
End Collection	End Collection C0

Figure 5 – Bottom Descriptor Window

5. Then in the window above the Descriptor window, we'll add three strings:

String/LANGID	Value
String Descriptors	USBFS
LANGID	English (United States)
String	Gamepad
String	0003
String	Cypress

Figure 6 – String/LANGID window

6. Then in the window above the String/LANGID window,
 - a. Set the *Vendor* and *Product ID* (any number is fine).
 - b. Set the *Manufacturer String* to Cypress.
 - c. Set the *Product String* to Gamepad.
 - d. Set the *Serial Number* string to 0003.
 - e. Then *Configuration Descriptor* >> *Interface Descriptor* >> *Class* set to HID.
 - f. *Configuration Descriptor* >> *HID Class Descriptor* >> *HID Report* set to GAMEPAD_RPT.

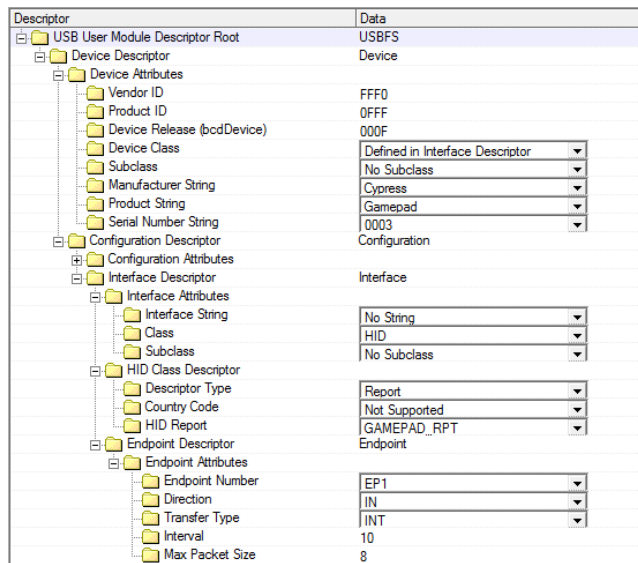


Figure 7 – Top Descriptor Window

That's it for the USBFS module! But before we can begin writing code, we must also configure the pins and global resources.

7. We will connect the buttons to P3[0] and P3[1] with pull-down resistance. Also, the analog outputs of the joystick will be connected to P0[1] and P0[7] and we will select these pins as AnalogInput.

Name	Port	Select	Drive	Interrupt
Port_0_0	P0[0]	StdCPU	High Z An	DisableInt
X_IN	P0[1]	AnalogInput	High Z An	DisableInt
Port_0_2	P0[2]	StdCPU	High Z An	DisableInt
Port_0_3	P0[3]	StdCPU	High Z An	DisableInt
Port_0_4	P0[4]	StdCPU	High Z An	DisableInt
Port_0_5	P0[5]	StdCPU	High Z An	DisableInt
Port_0_6	P0[6]	StdCPU	High Z An	DisableInt
Port_0_7	P0[7]	AnalogInput	High Z An	DisableInt
Port_1_0	P1[0]	StdCPU	High Z An	DisableInt
Port_2_7	P2[7]	StdCPU	High Z An	DisableInt
Button_1	P3[0]	StdCPU	Pull Down	DisableInt
Button_2	P3[1]	StdCPU	Pull Down	DisableInt
Port_3_2	P3[2]	StdCPU	High Z An	DisableInt

Figure 8 – Pin Configuration

8. After this is done, we will set the global resources in the top left window. Set *VC1* to 8 and *Ref Mux* to (Vdd/2)+/(Vdd/2). These are set to allow the ADC to sample using interrupts.

Global Resources	Value
Power Setting [Vcc / SysClk	5.0V / 24MHz
CPU_Clock	SysClk/8
Sleep_Timer	512_Hz
VC1= SysClk/N	8
VC2= VC1/N	1
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
Trip Voltage [LVD]	4.81V
LVDThrottleBack	Disable
Watchdog Enable	Disable

Figure 9 – Global Resources Configuration

Before we go on, let's take a longer look at the HID report we've defined and understand what is going on there.

HID

When the gamepad we will be making is connected to the PC, the PC will automatically know what to do with it and no set up will be needed because we will be defining an HID class for the device is PSoc Designer.

1.	USAGE_PAGE (Generic Desktop)	05 01
2.	USAGE (Game Pad)	09 05
3.	COLLECTION (Application)	A1 01
4.	USAGE (Pointer)	09 01
5.	COLLECTION (Physical)	A1 00
6.	USAGE (X)	09 30
7.	USAGE (Y)	09 31
8.	LOGICAL_MINIMUM (0)	15 00
9.	LOGICAL_MAXIMUM (255)	26 FF 00
10.	PHYSICAL_MINIMUM (0)	35 00
11.	PHYSICAL_MAXIMUM (1023)	46 FF 03
12.	REPORT_SIZE (10)	75 10
13.	REPORT_COUNT (2)	95 02
14.	INPUT (Data,Var,Abs)	81 02
15.	END_COLLECTION	C0
16.	USAGE_PAGE (Button)	05 09
17.	USAGE_MINIMUM (Button 1)	19 01
18.	USAGE_MAXIMUM (Button 2)	29 02
19.	LOGICAL_MINIMUM (0)	15 00
20.	LOGICAL_MAXIMUM (1)	25 01
21.	REPORT_SIZE (1)	75 01
22.	REPORT_COUNT (2)	95 02
23.	INPUT (Data,Var,Abs)	81 02
24.	REPORT_SIZE (1)	75 01
25.	REPORT_COUNT (6)	95 04
26.	INPUT (Cnst,Var,Abs)	81 03
27.	END_COLLECTION	C0

Figure 10 – HID Report

This HID report was defined using the HID Descriptor Tool found on www.usb.org as well as the HID Usage Tables found there. Let's go through an analysis of the report:

- Line-1 tells the host to look in usage page 01 in the HID usage tables, which is for generic desktop.
- Line-2 defines the device as a gamepad.
- Line-3 opens an application collection that will have the pointer information and the button information.
- Line-4 tells that the next collection will be a pointer.
- Line-5 opens another physical collection which will group the pointer information: X and Y axes.
- Line-6 and Line-7 define the X and Y axes.
- Line-8 and Line-9 define that the logical minimum and logical maximum for the axes will be 0 and 255 respectively.

- Line-10 and Line-11 define that the physical minimum and physical maximum for the axes will be 0 and 1023 respectively. This is because we are using a 10-bit ADC to sample from the analog X, Y values of the joystick. The highest value that 10 bits can represent is 1023.
- Line-12 and Line-13 define that each axis is represented by 16 bits of data and there are 4 such data present in the collection.
- Line-14 tells that all the data present in this collection will be a part of an input report.
- Line-15 closes the collection of pointers.
- Line-16 defines that the following information will be from usage page 09 which is the buttons page.
- Line-17 and Line-18 define that Button 1 to Button 2 in the usage page will be used.
- Line-19 and Line-20 define that the logical minimum and logical maximum for the buttons will be 0 (not pressed) and 1 (pressed).
- Line-21 and Line-22 define that each button will be represented by a single bit and there will be 2 buttons represented.
- Line-25 defines that the buttons will be a part of an input report. Any report should be built with bit numbers in multiples of 8. In our buttons report we have 2 bits. So, we need to pad the report with 6 constant bits. This is done by Line-24 to Line-26.
- Line-27 closes the complete collection and is the end of report.

Now we are done configuring the hardware and we may begin writing the code! From the *Device Editor* view that we are currently at, click on the *Generate Application* button and then switch over to the *Application Editor* view. Refer to the Appendix for the code to add to main.c. After adding the code, we will now build it and program it into the microcontroller.

Plug it in!

Now that the program is loaded into the controller, plug the USB cable to the USB-MINI B connector on the board. If using the CY3214 board, make sure there is a jumper on VBUS to power board via USB. Once connected, the device will enumerate as “Gamepad” or whatever you called your device and Windows will automatically configure the driver. Check out how the gamepad works by going to *Control Panel >> Game Controllers*. Then, double click on your device. This will open the properties window which is populated with the number of buttons and axes according to the HID report we defined during the USB Setup Wizard of the USBFS module. Make sure to calibrate the controller when using it for the first time.

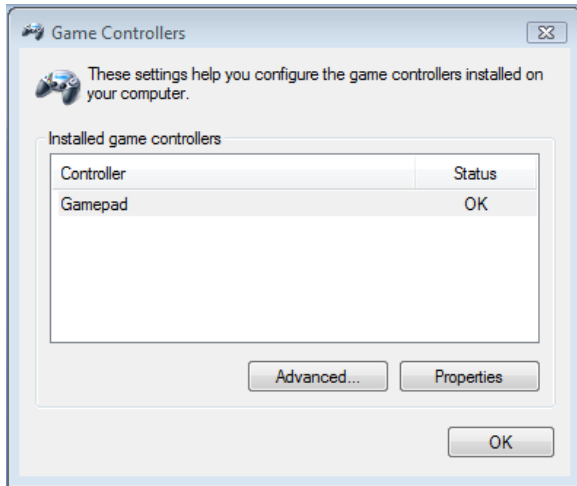


Figure 11 – Game Controllers Window

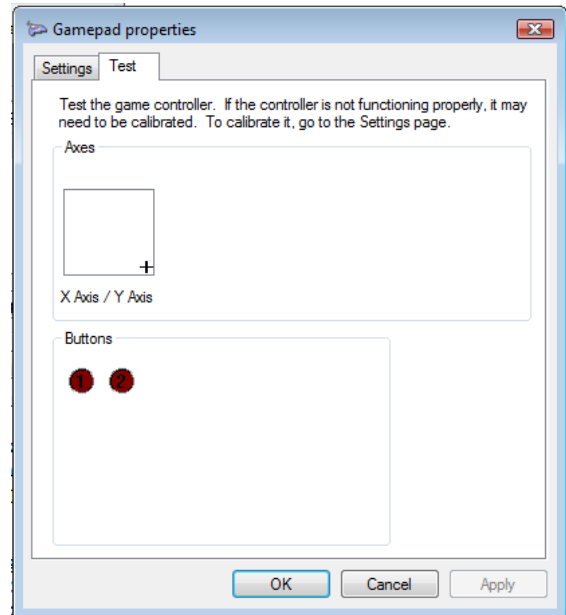


Figure 12 – Gamepad Properties

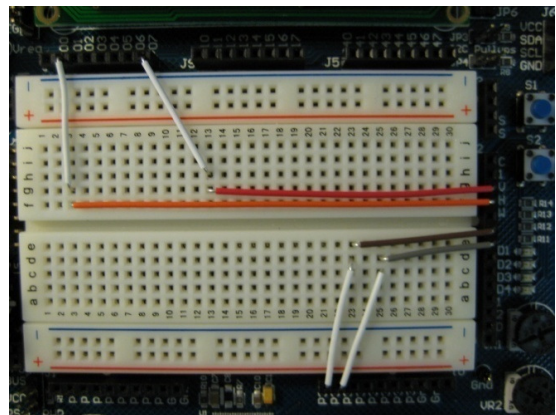


Figure 13 – Hardware Configuration

Appendix SOURCE CODE

```
//-----  
// C main line  
//-----  
  
#include <m8c.h>           // part specific constants and macros  
#include "PSoCAPI.h"      // PSoC API definitions for all User Modules  
  
int XAxis;  
int YAxis;  
unsigned char Buttons;  
unsigned char DataChangeFlag;  
  
int ConvertSmallEndian(int BigEndian);  
  
struct {  
    int XAxis; // stores X axis value  
    int YAxis; // stores Y axis value  
    unsigned char Buttons; // stores button states  
} GamePadReport;  
  
void main(){  
    unsigned char i;  
    M8C_EnableGInt; // enable global interrupts  
  
    // start the user modules  
    PGA_Start(PGA_HIGHPOWER);  
    ADC_Start(ADC_HIGHPOWER);  
    USBFS_Start(0, USB_5V_OPERATION);  
    while(!USBFS_bGetConfiguration()); // wait for enumeration to complete  
  
    USBFS_LoadInEP(1, (char*)&GamePadReport, sizeof(GamePadReport), USB_NO_TOGGLE);  
  
    while(1){  
        // Enable Pull down resistors and read the Buttons from Port3  
        PRT3DR &= ~0x03;  
        Buttons = PRT3DR;  
  
        // Connect P0[1] to input of X value of joystick and do a single ADC Conversion  
        // AMX_IN = register that controls the analog muxes that feed signals in from port pins  
        // into the analog column. Refer to TRM for more info.  
        AMX_IN &= ~0x03;  
        ADC_GetSamples(1);  
        while(ADC_fIsDataAvailable() == 0);  
        XAxis = ConvertSmallEndian(ADC_wClearFlagGetData());  
  
        // Connect P0[7] to input of Y value of joystick and do a single ADC Conversion  
        AMX_IN &= ~0x03;  
        AMX_IN |= 0x03;  
        ADC_GetSamples(1);  
        while(ADC_fIsDataAvailable() == 0);  
        YAxis = ConvertSmallEndian(ADC_wClearFlagGetData());  
  
        // check to see if anything has changed  
        if(Buttons != GamePadReport.Buttons)  
            DataChangeFlag = 1;  
        if(XAxis != GamePadReport.XAxis)  
            DataChangeFlag = 1;  
        if(YAxis != GamePadReport.YAxis)  
            DataChangeFlag = 1;  
  
        if(DataChangeFlag){  
            DataChangeFlag = 0;  
            GamePadReport.Buttons = Buttons;  
            GamePadReport.XAxis = XAxis;  
            GamePadReport.YAxis = YAxis;  
            while(!USBFS_bGetEPackState(1));  
            USBFS_LoadInEP(1, (char*)&GamePadReport, sizeof(GamePadReport), USB_TOGGLE);  
        }  
    }  
}
```

```
    }  
}  
  
// This function converts the big endian number format used by  
// PSoC compiler to small endian format of USB standard  
int ConvertSmallEndian(int BigEndian){  
    int Temp;  
    Temp = (BigEndian >> 8) & 0x00FF;  
    Temp |= ((BigEndian << 8) & 0xFF00);  
    return Temp;  
}
```

References:

1. PSoC Mixed Signal Array Technical Reference Manual version 2.1
2. Module Data Sheets (when selecting the component in PSoC Designer)
3. HID Usage Tables
4. Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID)
5. M. Ganesh Raaja
6. Professor Rajesh Gupta