USB Interrupt Transfer Example – PSoC® 3 / PSoC 5

Project Objective

This code example demonstrates how to perform **"USB Interrupt Transfer"** from a PC using the USB HID driver and PSoC® 3 device.

Overview

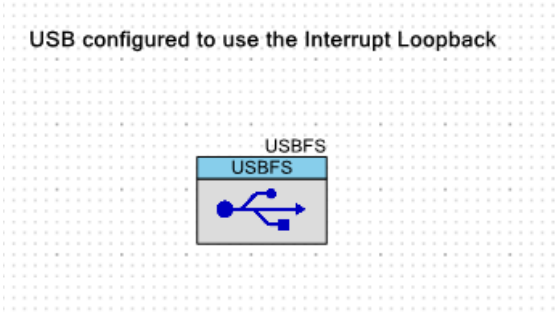
USB interrupt transfer is a transfer type where in the USB host interrupts the USB device at regular interval and performs an USB transfer. This interval is selected by the user and a guaranteed data transfer occurs during this interval.

A full speed USB component in PSoC 3 / PSoC 5 is used in this project that enumerates as a generic HID class device. This project transfers 64 or lesser number of bytes of data from the PC to the PSoC 3 / PSoC 5 device using the USB interrupt 'Out endpoint'. The sent data is modified by the PSoC 3 / PSoC 5 device and is sent back to the PC using interrupt 'In endpoint'.

A GUI is created using C# for sending data from the PC to the PSoC 3 / PSoC 5 device and to read back the data from the PSoC 3 / PSoC 5 device. For other related USB Examples and resources scroll to the Section, **Error! Reference source not found.**, that is at the end.

Top Design

The following figure illustrates the components and their routing:



The following figure shows pin placement (as in .cydwr file).

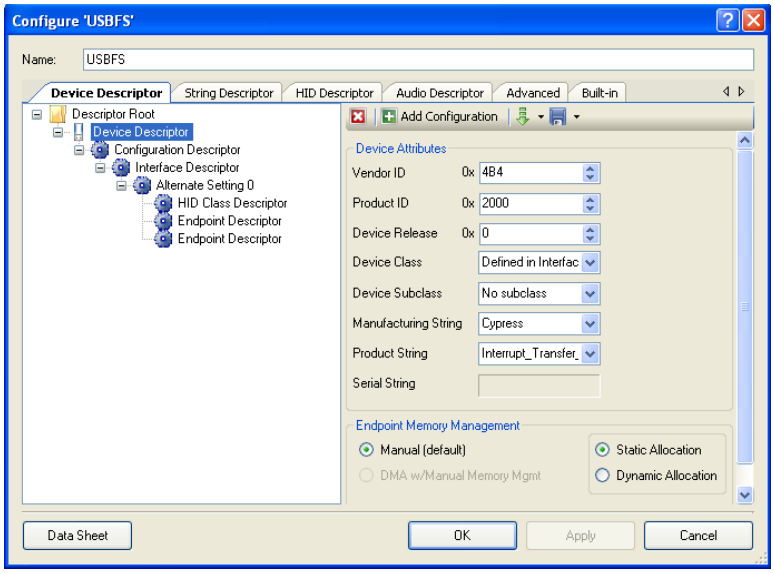
Alias	Name	Pin	Lock
	\USBFS:Dm\	P15[7]	☑
	\USBFS:Dp\	P15[6]	☑

The USB uses the fixed pins, P15 [6] and P15 [7]. After the USB is placed on the TopDesign, the pins are automatically blocked.

Component Configuration

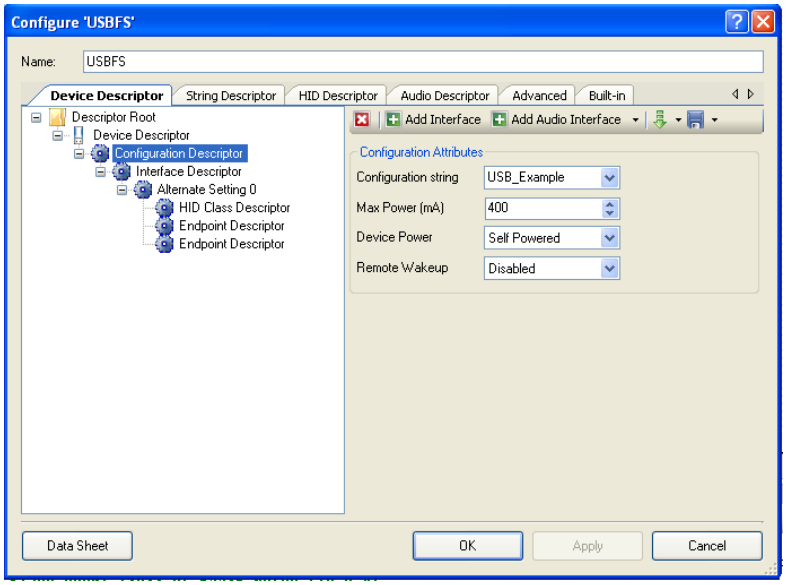
To configure a full speed USB component, right click on the component and select configure (or double click on the component). This opens up a configuration wizard with various configurable parameters for USB as shown from Figure 1 to Figure 6. The following figures show the component parameter settings for the USBFS component used in the project.

Figure 1. Device Descriptor



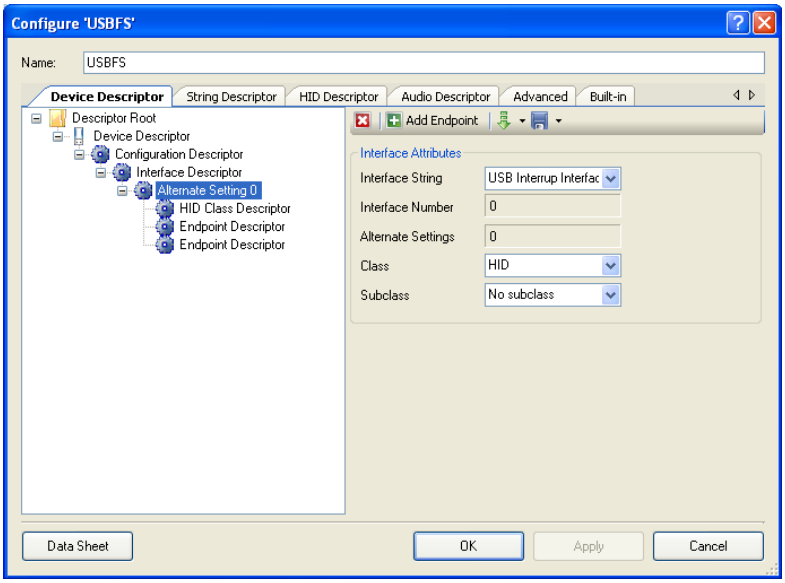
In the figure, the String Cypress and Interrupt_Transfer_Example added to the device Descriptor are automatically updated in the **String** descriptor tab.

Figure 2. Configuration Descriptor



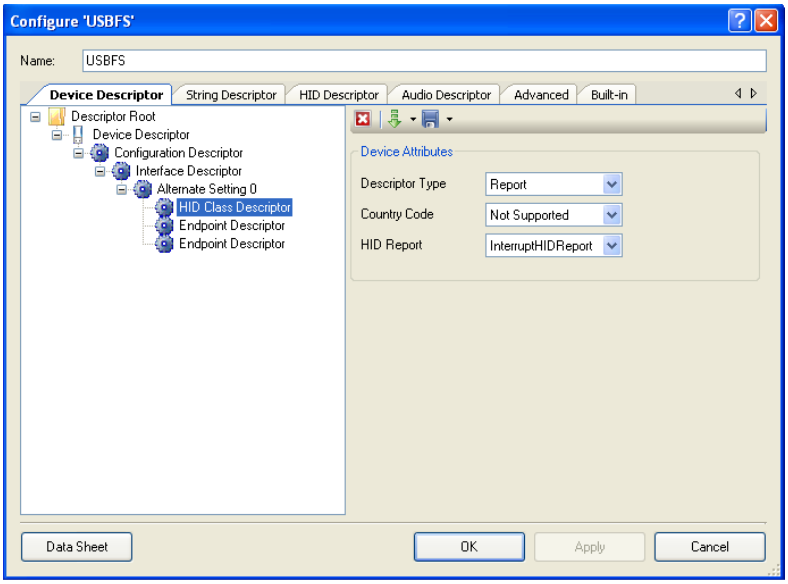
In the figure, the String USB_Example is automatically updated in the **String** descriptor tab.

Figure 3. Interface Descriptor



In the figure, the String USB Interrupt Interface is automatically updated in the **String** descriptor tab.

Figure 4. HID Class Descriptor



In the figure, the HID Report InterruptHIDReport is created in the **HID** descriptor tab.

The following sections describe the creation of a HID report.

Figure 5. In Endpoint Descriptor

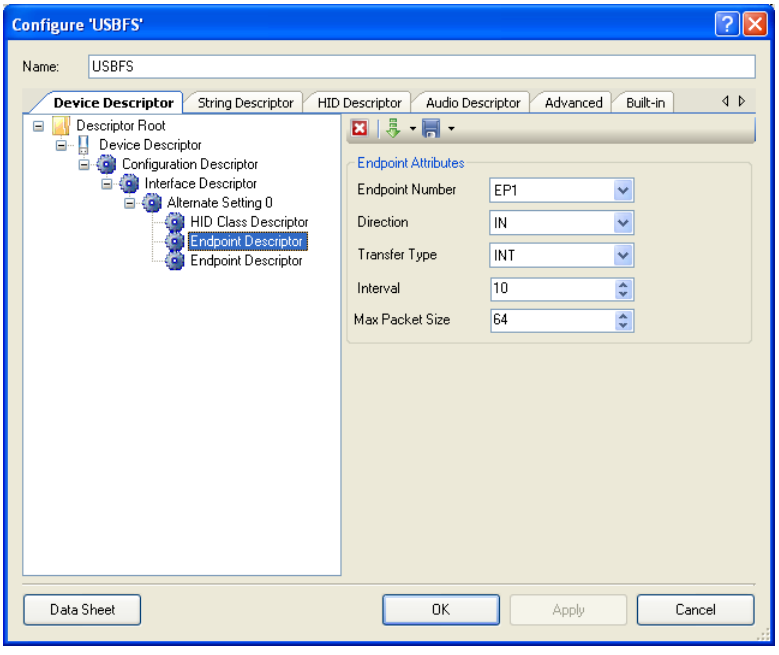
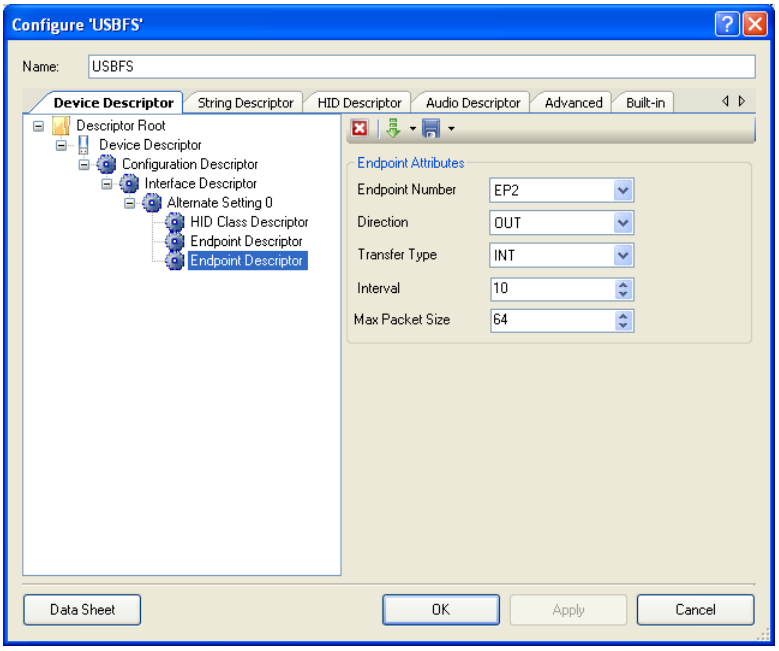


Figure 6. Out Endpoint Descriptor

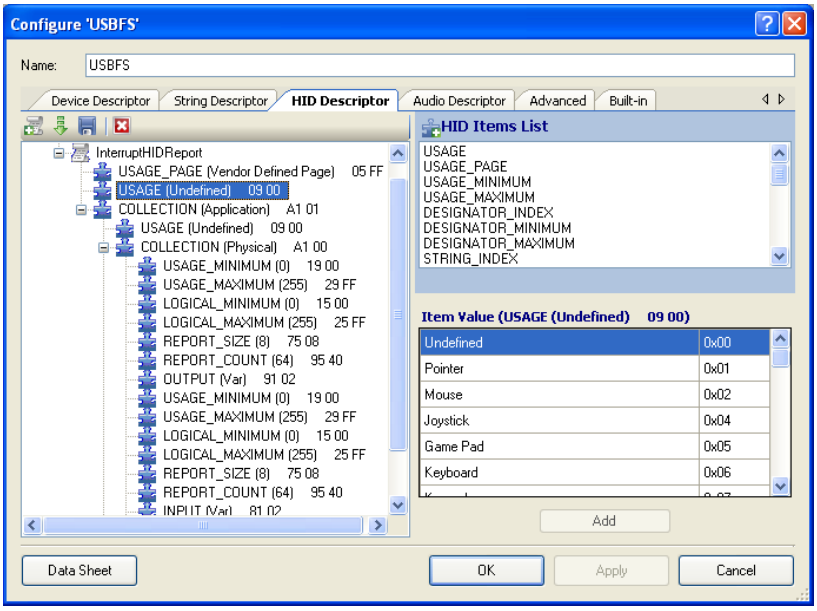


The PSoC 3 / PSoC 5 device is configured as a generic HID device with one Out Interrupt endpoint and one In Interrupt endpoint, each having a data buffer size of 64 bytes (see Figure 5 and Figure 6). On every scanning interval (10 ms as specified by the endpoint configuration) the host checks for any In or Out packet from or to the HID device and performs an In or Out operation. When an Out data packet is sent to the device using the GUI, an Out endpoint interrupt is raised at the PSoC 3 / PSoC 5 device. When the interrupt occurs, the received data bytes are read and the same data is incremented and loaded back to the In Interrupt endpoint. This In Endpoint data is read by the PC when the user performs data read operation from the GUI (The details of the GUI are discussed in later section). The Usage page for the HID is defined as vendor specific.

The In and Out reports are defined in the HID descriptors; the logical minimum and logical maximum are fixed to 0 and 255 respectively. The same values are also used for usage minimum and usage maximum. An In report and an Out report, each having a data size of 64 bytes, are declared in the HID descriptor.

To create a HID report, **HID Descriptor** tab in the USBFS configuration page should be selected. The HID descriptor is a set of HID Items. To add any items to the descriptor, select one of the HID items list (for example, 'USAGE') and choose any required value from the 'item Value' (for example, 'Pointer'). Then click **Add** to add this item to the report. The complete report can be seen on the left pane.

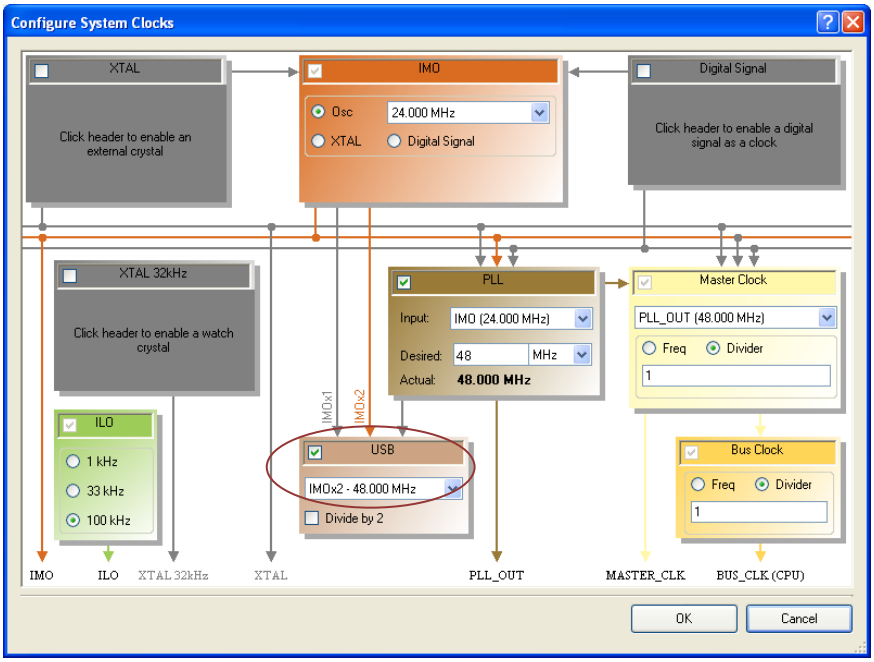
Figure 7. HID Report Descriptor



Design Wide Resources

The clock tree for the project is shown in Figure 8. (The clock tree for a project can be reached by double clicking on the design wide resource file (USB_Interrupt_Transfer_Example.cydwr file) in the workspace explorer and clicking on the **Clocks** tab. The clocks configuration wizard can be opened by double clicking on the clocks listed in the **Clocks** tab).

Figure 8. Clock Tree



The clock for the USB component is derived from IMO. The frequency of the IMO has to be 24 MHz. The D+ and D- pins are also mapped to the corresponding pins using the design wide resources window. In the current code

example, USB is declared as self powered from a 3.3 V source. The USB bus voltage is not used for any functionality. The other settings in the .cydwr file can be kept in the default state.

Operation

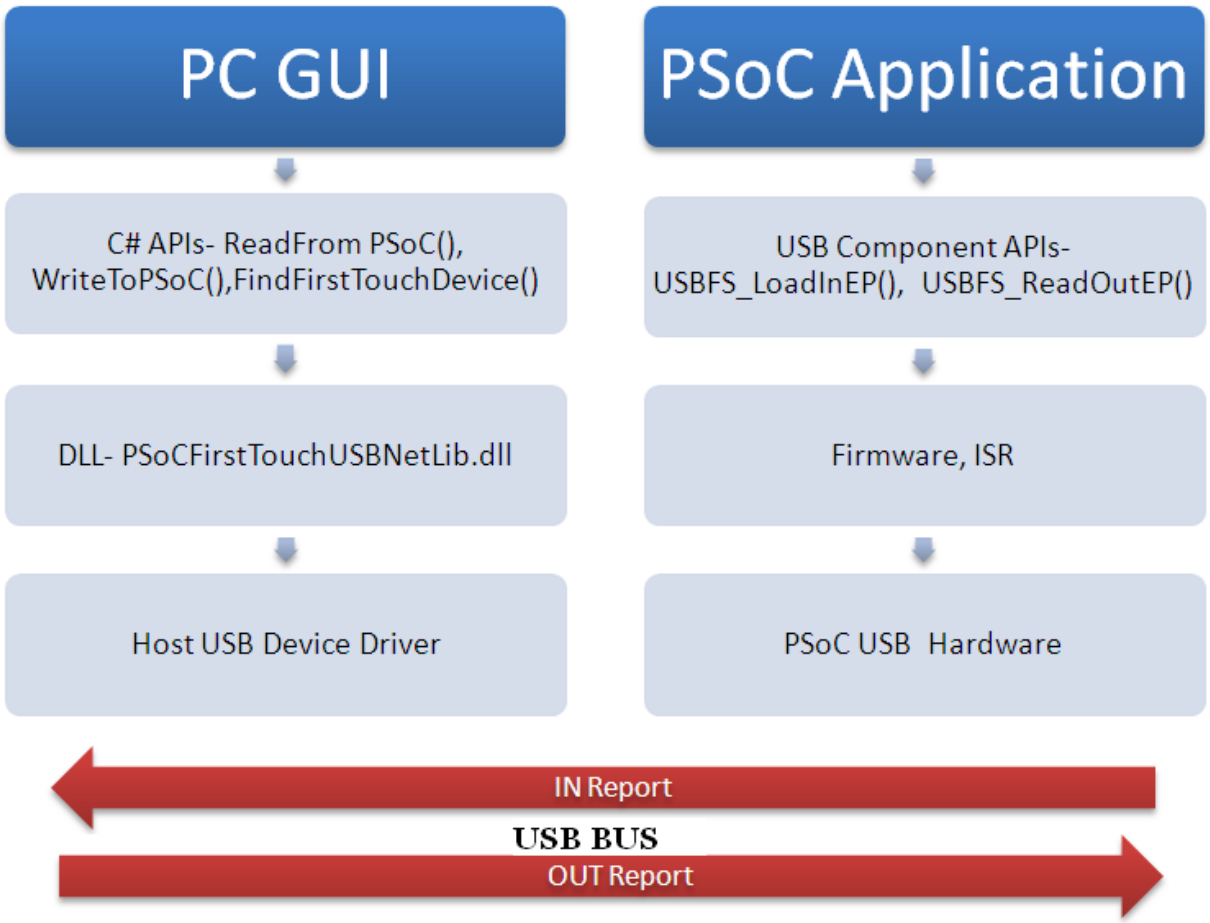
The main routine for this project is as follows.

1. The Global interrupts are enabled for USB operation
2. The USBFS user module is started at 3.3 V mode operation. (The USB can work from 3.3 V to 5 V)
3. After the user module initialization, the code waits for USB enumeration operation to be completed
4. After the enumeration completes, the Interrupt OutEndpoint is enabled for receiving data from the PC
5. When the Out endpoint interrupt occurs, the received data from PC is incremented by one and the In endpoint is filled with the incremented data
6. The Read request can be sent from the GUI by clicking on the Read button; the In endpoint data will be sent to the PC on the read request

GUI

Figure 9 shows the complete implementation of the project, that includes the GUI on the PC and the PSoC3 interaction.

Figure 9. Implementation of the project



For exchanging data between the PC and PSoC 3 / PSoC 5, a simple GUI is developed in C# that uses the *PSoCFirstTouchUSBNetLib.dll* file, which in turn uses the *CyUSB.sys* driver for communicating to the PSoC 3 / PSoC 5 device. The *dll* file, placed in folder `\GUI\Interrupt_Transfer_Example\bin\Release`, exposes two simple APIs, ReadFromPSoC and WriteToPSOC, which send and receive data to the PSoC device.

- byte[] ReadFromPSoC(): The API ReadFromPSoC() returns a byte array containing the input report from the PSoC HID device.

  Example: `PSoCData = gFirstTouchDevice.ReadFromPSoC();`

- WriteToPSoC(byte[] PSoCData): The API WriteToPSoC() sends a byte array as an OUT report to the PSoC HID device.

  Example: `gFirstTouchDevice.WriteToPSoC(PSoCData);`

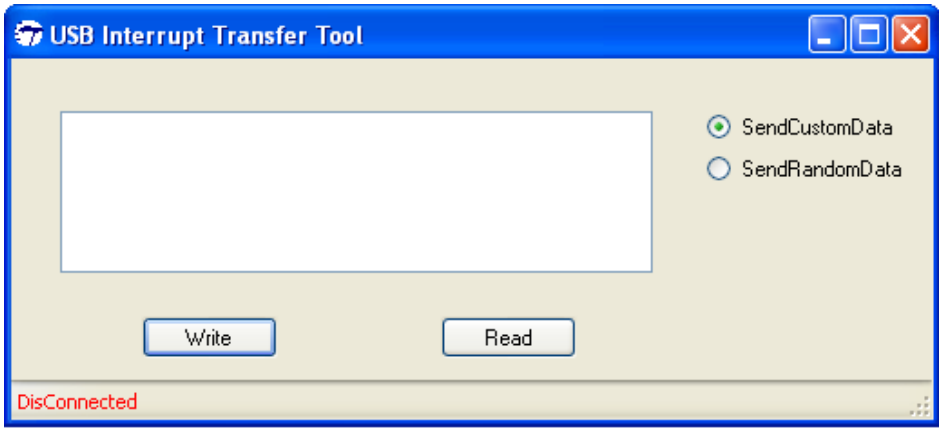
The PSoC device is acquired by the GUI application depending on the VID and PID of the device by using the method `FindFirstTouchDevice()`.

- `FindFirstTouchDevice(int VendorId, int ProductId)` : The API returns a handle to a connected USB device with the Vendor ID and Product ID specified as its input arguments.

  Example: `gFirstTouchDevice = FirstTouchHidDevice.FindFirstTouchDevice(0x4b4, 0xF100);`

To open the GUI click on the *Interrupt_Transfer_Example.exe* file, placed in the folder `\GUI\Interrupt_Transfer_Example \bin\Release`.

Figure 10. Snapshot of the GUI

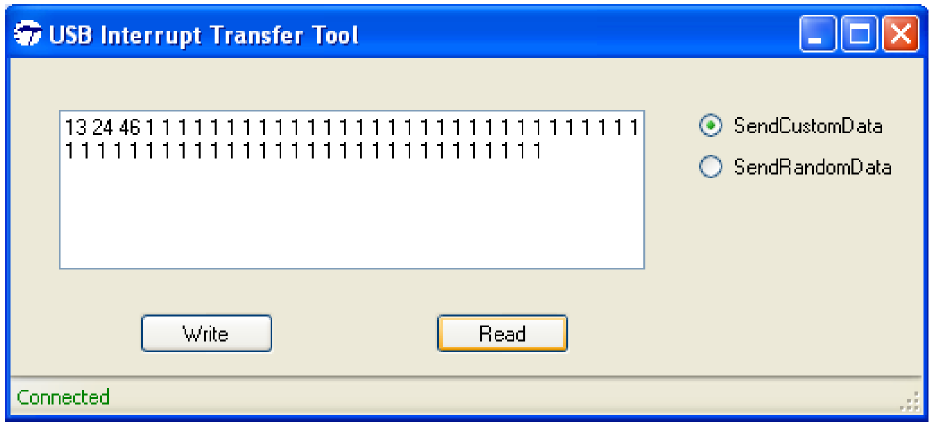


When the USB device (PSoC 3 / PSoC 5) is programmed and connected to PC, the GUI task bar shows Connected at the bottom left corner. When the device is not connected or when it is not working properly as a USB device, the task bar shows as 'disconnected'. Users can send custom data by clicking the "SendCustomData" radio button. A predetermined sequence of 64 bytes can be sent by selecting the "SendRandomData" button. When the "SendCustomData" option is selected, the data to be sent is entered in the text box and when the **Write** button is clicked, 64 bytes of the data inside the text box (zero is inserted if all the data is not present) is sent to the PSoC 3 / PSoC 5 (that is, the Out endpoint in PSoC 3 / PSoC 5 receives this data). This operation triggers out endpoint interrupt inside the PSoC 3 / PSoC 5 device. This interrupt routes the program to `CY_ISR(USBFS_EP_2_lSR)` inside the *USBFS_episr.c* file.

In the ISR:

1. A flag to indicate the occurrence of interrupt is set (USB_interruptFlag)
2. In the main code, when this flag is set, the function ProcessEP2Data is called
3. The function ProcessEP2Data does the following:
   a. Reads how many bytes have been sent to the Out endpoint
   b. Reads the bytes sent to the Out endpoint and stores to a local buffer
   c. Increments the received data
   d. Puts the incremented data to the In endpoint buffer so that when the **Read** button is clicked in the GUI, the data from the In endpoint buffer is sent to PC

Figure 11. Snapshot of the Loopback Data



Note The read operation must be performed only after the data is written to the device. This is because the firmware is written to act in such a manner.

Once an OUT transfer is made by clicking the Write button, this data is received by PSoC, incremented and loaded to the In Endpoint. The USB HID implementation in the PC checks every 10 ms to see if there is In Endpoint data from PSoC. Once the In Endpoint is loaded with data, during the next request from the PC, this data is read. This happens whether or not the Read button is pressed. Later when the Read Button is pressed the data read during the previous successful In Endpoint transfer is displayed on the GUI.

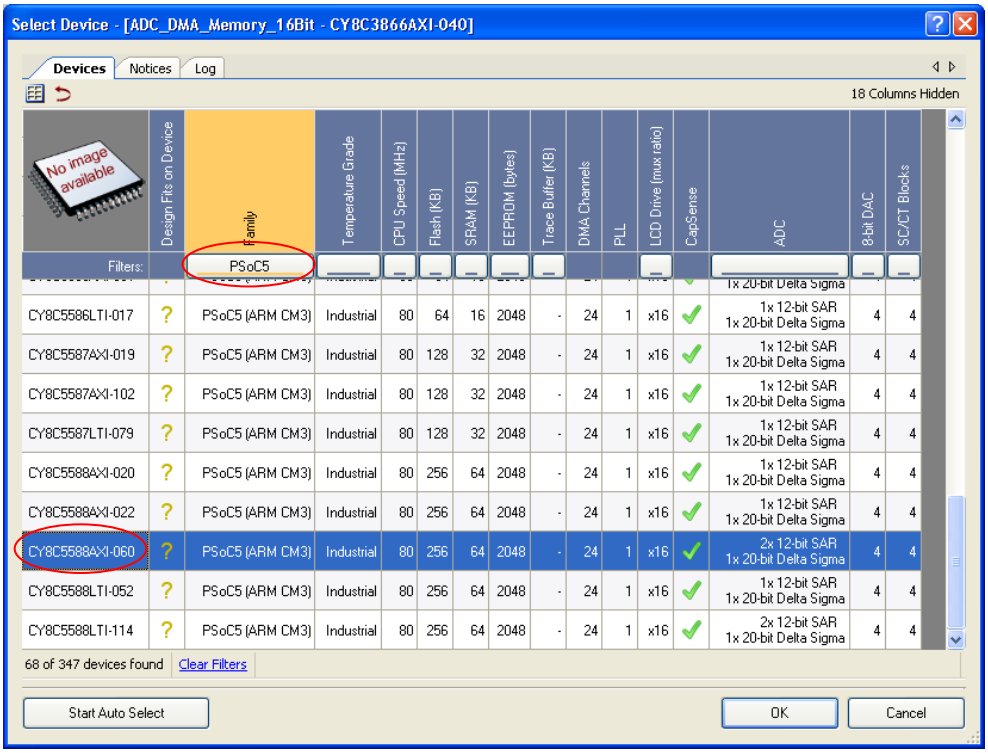
Hardware Connections

- The project is tested with the PSoC Development Kit CY8CKIT-001
- The kit should be used with the default state of the jumpers. Refer CY8CKIT-001 PSoC Development Kit Guide
- The DVK switch supply (SW3) should be set to 3.3 V
- If the USB should be run at 5 V, change the SW3 position to 5 V
- In the main.c, initialize the USB to run at 5 V. Use the API `USBFS_Start` with parameter `USBFS_5V_OPERATION`
- Connect a USB cable from the DVK USB port to PC USB port.

Output

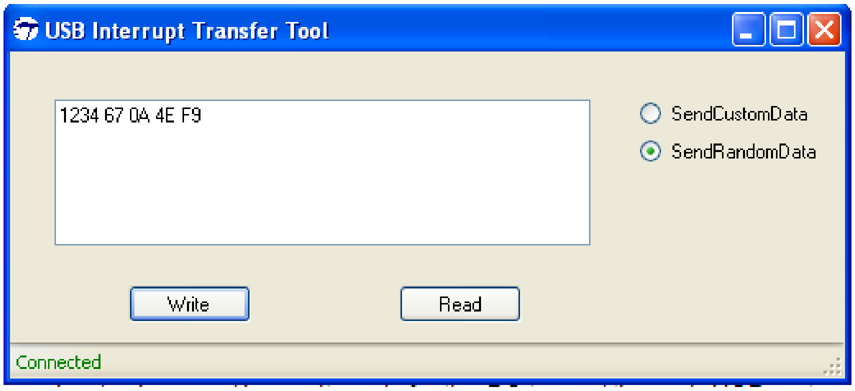
- Build the project and program the chip.
  Note The default device selection is PSoC 3 (CY8C3866AXI-040).To use this project with PSoC 5 family, do the following:
- Go to **Project → Device Selector →** Select **PSoC 5** device (CY8C5588AXI-060), build the project again and program the PSoC 5 device as follows:

Figure 12. Device Selector



- Reset the device by pressing the SW4 (Reset Switch).
- The device enumerates and gets automatically bound to HID driver. (The PC pops up a window at the Taskbar mentioning that 'A New Hardware was found and Is ready to use'
- Open the PC GUI software provided with this code example (located in the code example folder in GUI\Interrupt_Transfer_Example \bin\Release\ Interrupt_Transfer_Example.exe).
- Enter up to 64 byte of data in hexadecimal number format with or without spaces between the numbers and click the **Write** button. The GUI parses the data and sends it accordingly.

Figure 13. USB Interrupt Transfer Tool – Write Operation.



- The GUI sends 64 bytes of hexadecimal value to the target through USB port. The GUI will pop up saying 'Data Sent'
- It automatically appends 0s if the user enters less than 64 bytes. The PSoC 3 / PSoC 5 device receives 64 bytes and increments the received values and keeps it ready for the PC to read through USB port.
- Click the **Read** button, the 64 bytes of data read from PSoC 3 are displayed on the GUI. The read data are write data incremented by 1 as shown below.

Figure 14. USB Interrupt Transfer Tool – Read Operation