Мікропроцесорна техніка

(лекція 11)
Благітко Б.Я.
2019 р.

PSoC Creator 4.2
Designing with PSoC 3/5

PSoC@3/5
USB

PSoC Creator 4.2
Designing with PSoC 3/5

PSoC@3/5

Передача даних від мікроконтролера PSoC 3/5 до ЕОМ, і навпаки, за допомогою шини передачі даних USB

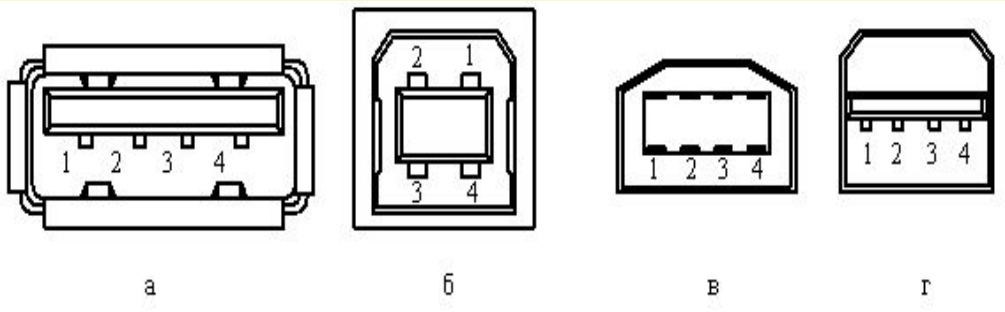
PSoC Creator 4.2
Designing with PSoC 3/5

PSoC@3/5
USBFS
(Full Speed USB)

PSoC Creator 4.2
Designing with PSoC 3/5

Універсальна послідовна шина USB

- USB (Universal Serial Bus – універсальна послідовна шина) є промисловим стандартом розширення архітектури ПЕОМ, орієнтованим на інтеграцію з телефонією і пристроями побутової електроніки.



Гнізда USB: а – тип «А», б – тип «В» стандартні, в, г – міні типу «В»

Універсальна послідовна шина USB

Призначення виводів розняття USB:

Контакт	Коло
1	VBus (+5B)
2	D–
3	D+
4	GND

Шина USB має три режими роботи:

- Низькошвидкісний (LS, low-speed)
- Повношвидкісний (FS, full-speed)
- Високошвидкісний (HS, high-speed)

Типи передавання даних USB

Архітектура USB допускає чотири базові типи передавання даних:

- "Посилки керування ” (control transfers) використовуються для конфігурування пристроїв під час їх під'єднання і для управління пристроями в процесі роботи.
  Протокол забезпечує гарантовану доставку даних.
- Передавання масивів даних (bulk data transfers) – це передавання без будь-яких зобов’язань щодо затримки доставки і швидкості передавання.
  Передавання масивів можуть займати всю смугу пропускання шини, вільну від передавання інших типів.
  Передавання масивів доречні для обміну даними з принтерами, сканерами, пристроями зберігання і тощо.

Типи передавання даних USB

- Переривання (interrupt) короткі передачі, які мають спонтанний характер і повинні обслуговуватись не повільніше, ніж того потребує пристрій.

  Переривання використовують, наприклад, для вводу символів з клавіатури або для передавання повідомлення про переміщення миші.
- Ізохронні передачі (isochronous transfers) неперервні передачі в реальному часі, які займають попередньо узгоджену частину пропускної здатності шини з гарантованим часом затримки доставки.

  Ізохронні передачі потрібні для потокових пристроїв: відеокамер, цифрових аудіопристроїв (колонки USB, мікрофон), пристроїв відтворення і запису аудіо- і відеоданих.

Кінцеві точки (endpoint)

Кінцева точка (endpoint) – це частина USB-пристрою, яка має унікальний ідентифікатор і є приймачем або відправником інформації, яка передається по шині USB.

Кожна кінцева точка має свій номер і описується такими параметрами:

- необхідна частота доступу до шини і допустимі затримки

обслуговування;

- необхідна смуга пропускання каналу;
- вимоги до обробки посилань;
- максимальні розміри переданих і прийнятих пакетів;
- тип передавання;
- напрям передавання (для передавання масивів і ізохронного обміну).

Дескриптори

Дескриптор пристрою (**device descriptor**) – це структура даних, або блок інформації, який позволяє хосту отримати опис USB-пристрою.

Стандартний дескриптор пристрою (standard device descriptor) містить основну інформацію про USB-пристрій в цілому і про всі існуючі конфігурації.

Уточнюючий дескриптор пристрою (device qualifier descriptor) містить додаткову інформацію про HS(high speed)-пристрій при його роботі на іншій швидкості.

Стандартний дескриптор конфігурації (standard configuration descriptor) містить інформацію про одну із можливих конфігурацій USB-пристрою.

Дескриптори

Стандартний дескриптор інтерфейсу (standard interface descriptor) містить інформацію про один із інтерфейсів, доступних при певній конфігурації USB-пристрою.

Стандартний дескриптор кінцевої точки (standard endpoint descriptor) – містить інформацію про одну із кінцевих точок, доступних при використанні певного інтерфейсу.

Дескриптор стрічки (Unicode string descriptor) – містить інформацію в форматі Unicode.

Дескриптори

HID дескриптори (human interface device descriptor) – дескриптори класу пристроїв USB для взаємодії з людиною. Цей клас включає в себе такі пристрої як клавіатура, мишка та ігровий контролер.

Аудіо дескриптори (audio descriptor) - дескриптори класу пристроїв USB для взаємодії аудіо пристроями. Цей клас включає в себе такі пристрої як навушники, мікрофон та колонки.

MIDI дескриптори (musical instrument digital interface descriptor) - дескриптори класу пристроїв USB для взаємодії музичними інструментами.

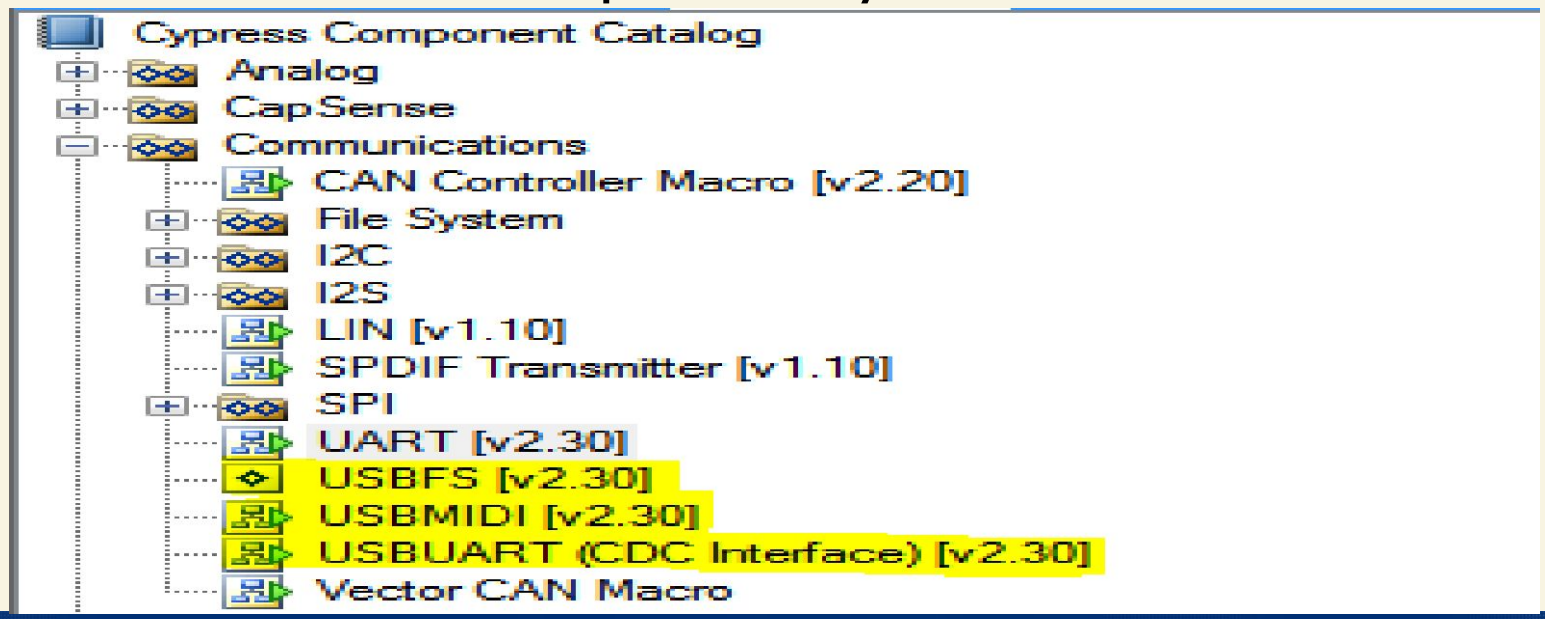
CDC дескриптори (communications device class descriptor) - дескриптори класу пристроїв USB для взаємодії з пристроями для мережі. Цей клас включає в себе такі пристрої як модеми та адаптери Ethernet.

USB

Компоненти в PSoC Creator 4.2, які працюють з USB

- USBFS
- USBMIDI
- USBUART

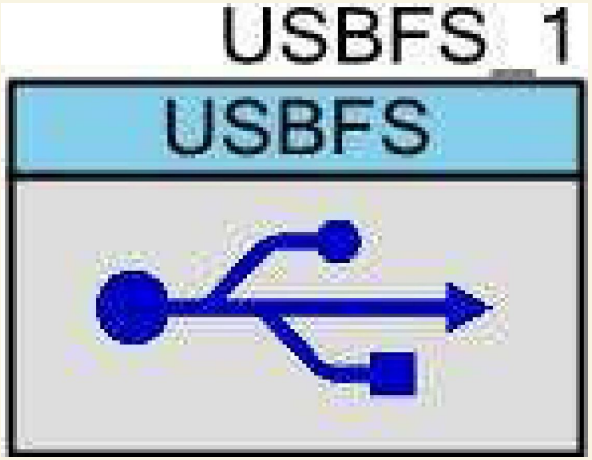
Ці компоненти знаходяться в компонент каталозі, в категорії “Комунікації”.



USBFS

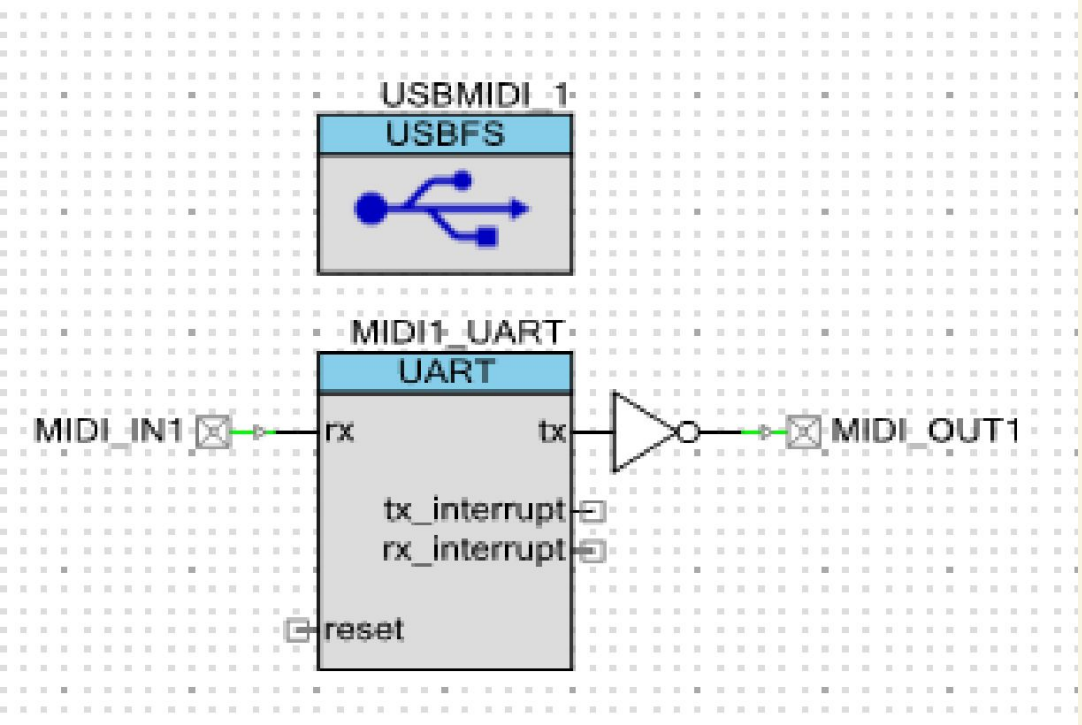
Компонента надає низькорівневий драйвер для контролю інцевої точки, яка декодує і відправляє запити від хоста USB.

Крім того, ця компонента забезпечує для користувача USBFS настроювач, який полегшує налаштування дескрипторів



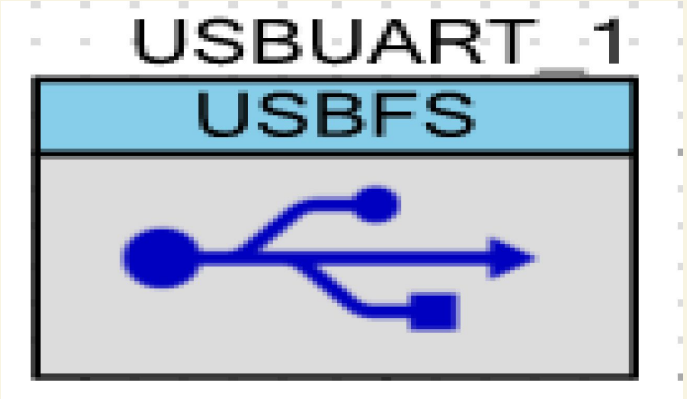
USBMIDI

USBFS MIDI забезпечує підтримку для комунікації із зовнішнім MIDI-обладнанням. Вона також забезпечує підтримку для класу USB пристрою визначеного для MIDI пристроїв.



USBUART

PSoC Creator каталог компонентів містить схеми реалізації макрос інтерфейсу CDC (також відомий як USBUART). Це USBFS компонента з дескрипторами налаштованими для реалізації інтерфейсу CDC. Це дозволяє використовувати CDC підтримку USBFS компоненти з мінімальною конфігурацією, яка потрібна.



When to Use a USBFS

Use the USBFS component when you want to provide your application with a USB 2.0 compliant device interface.

Quick Start

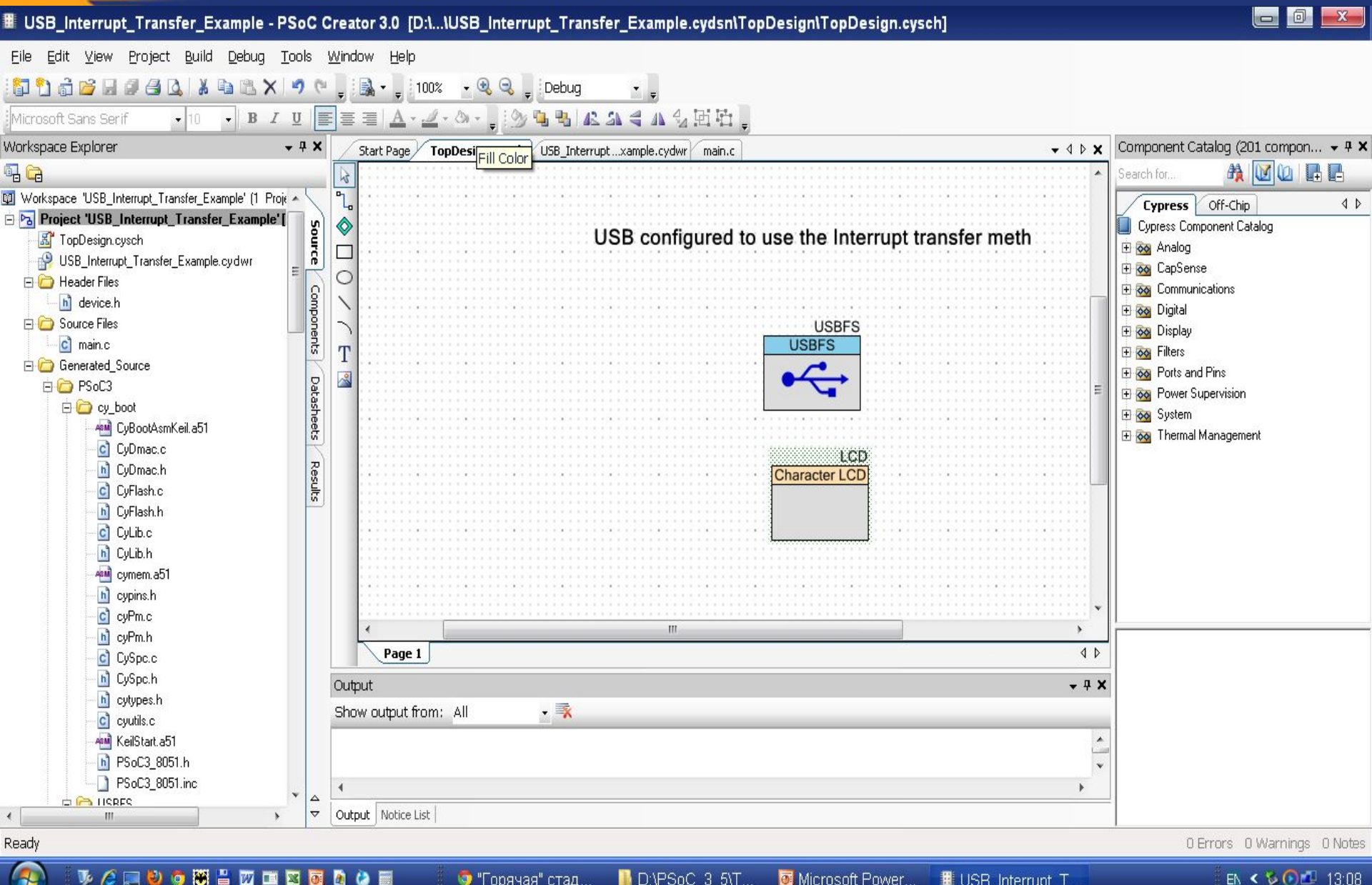
1. Drag a USBFS component from the Component Catalog onto your design.
2. Notice the clock errors in the Notice List window; double-click on an error to open the System Clock Editor.
3. Configure the following clocks for PSoC 3 :

i) ILO: Select 100 kHz.

ii) IMO: Select Osc 24.000 MHz.

i) USB: Enable and select IMOx2 – 48.000 MHz.

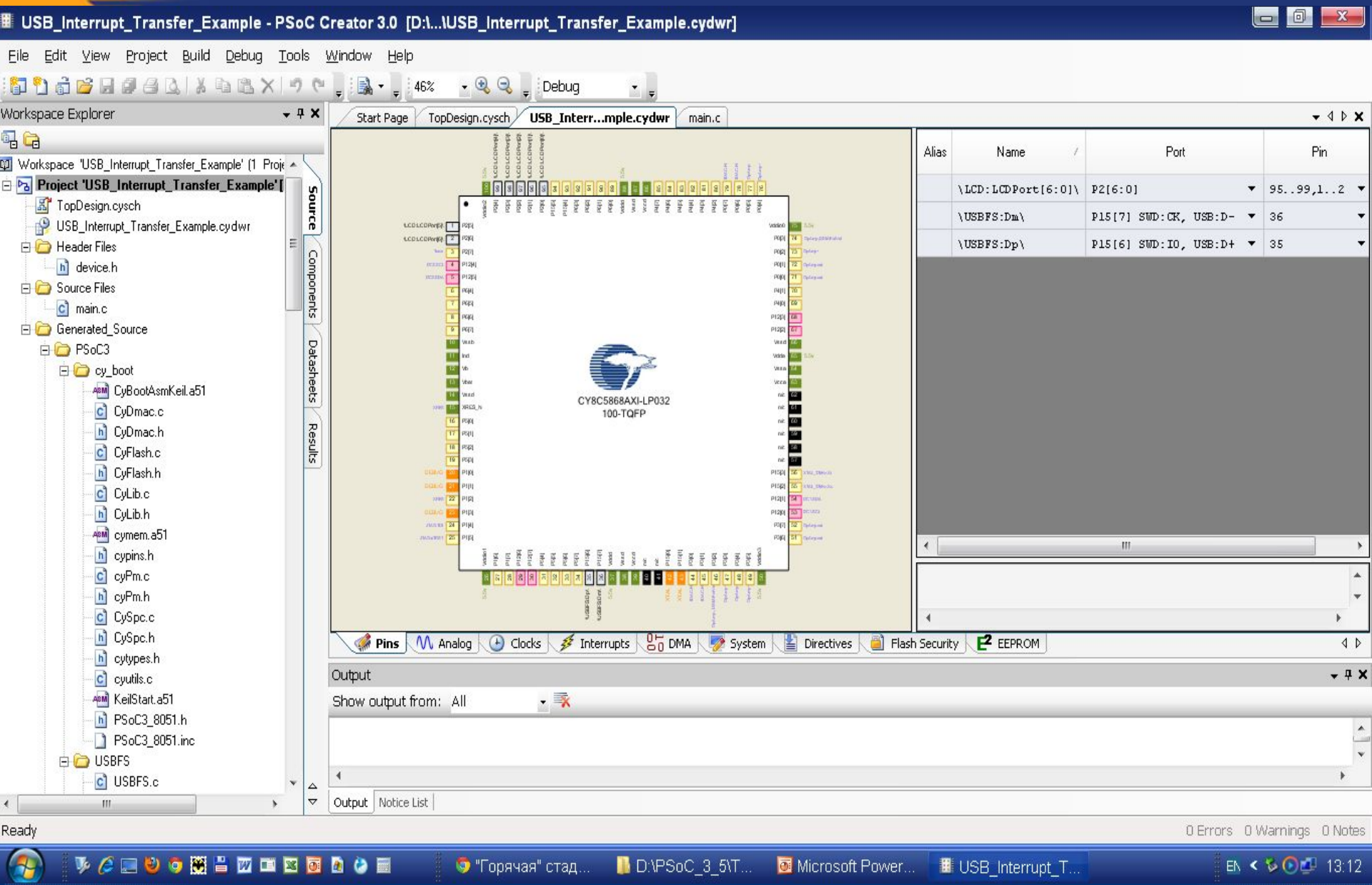
USBFS PSoC@3/5



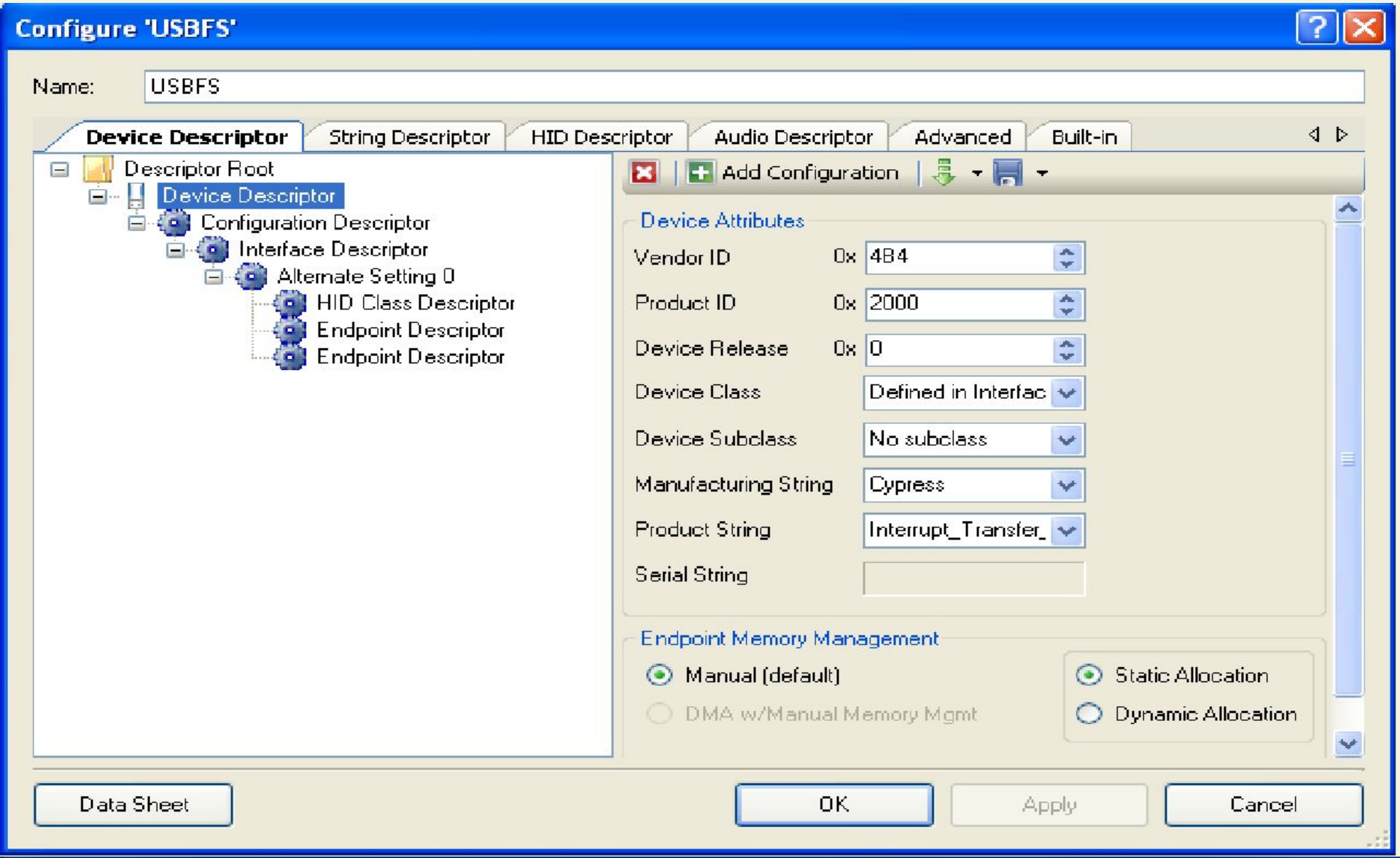
USBFS PSoC@3/5

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

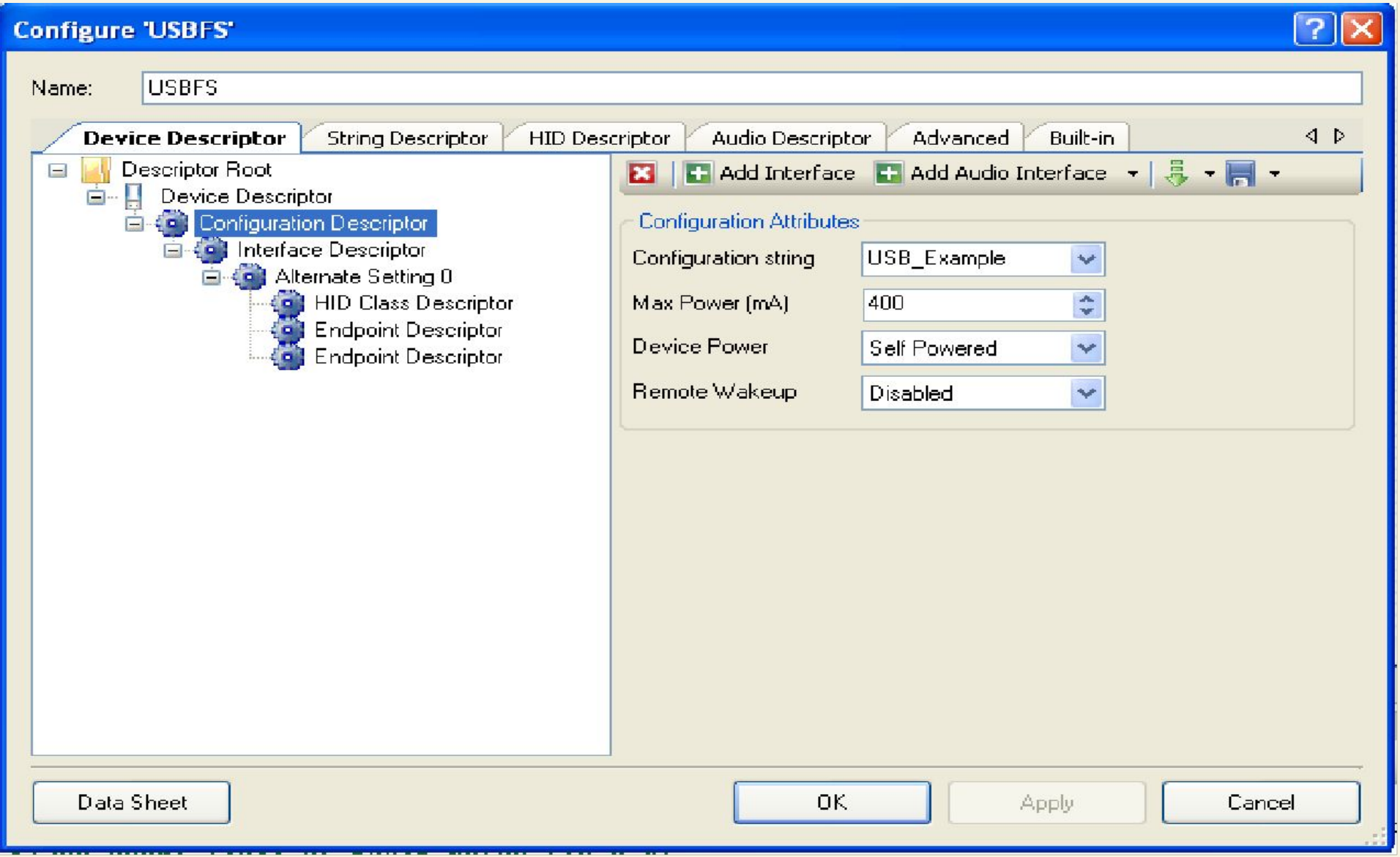
USBFS PSoC@3/5



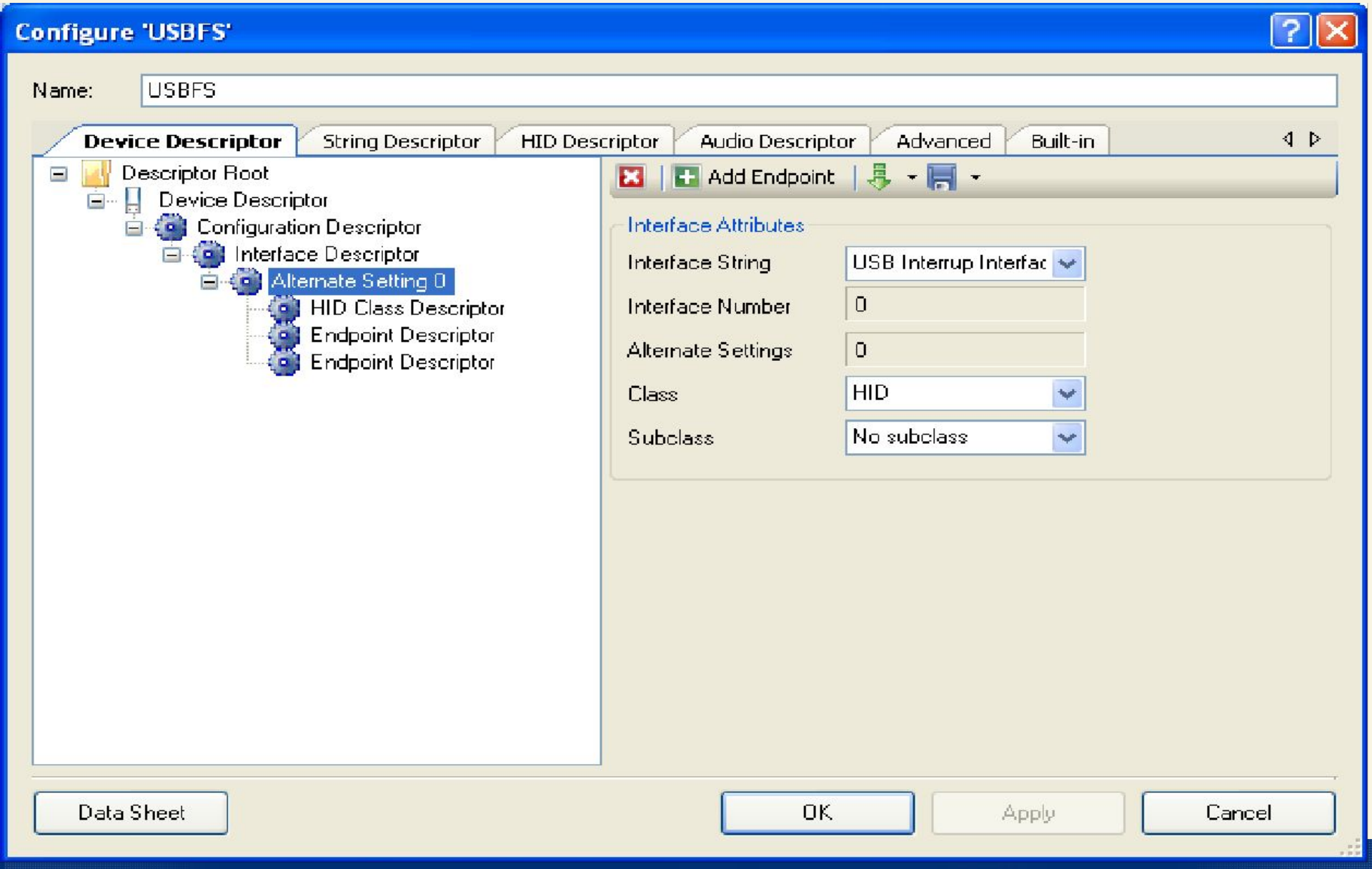
Device Descriptor



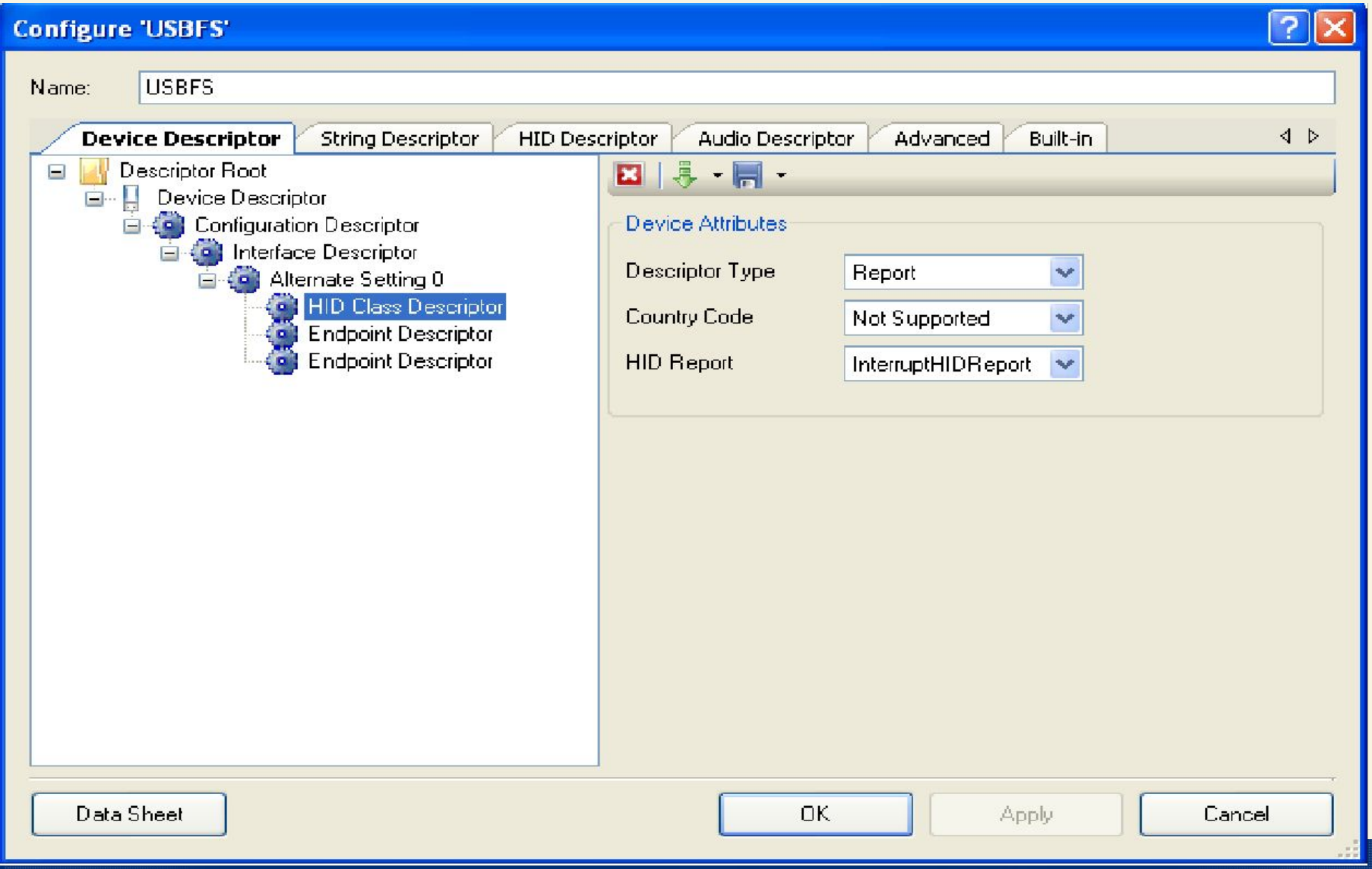
Configuration Descriptor



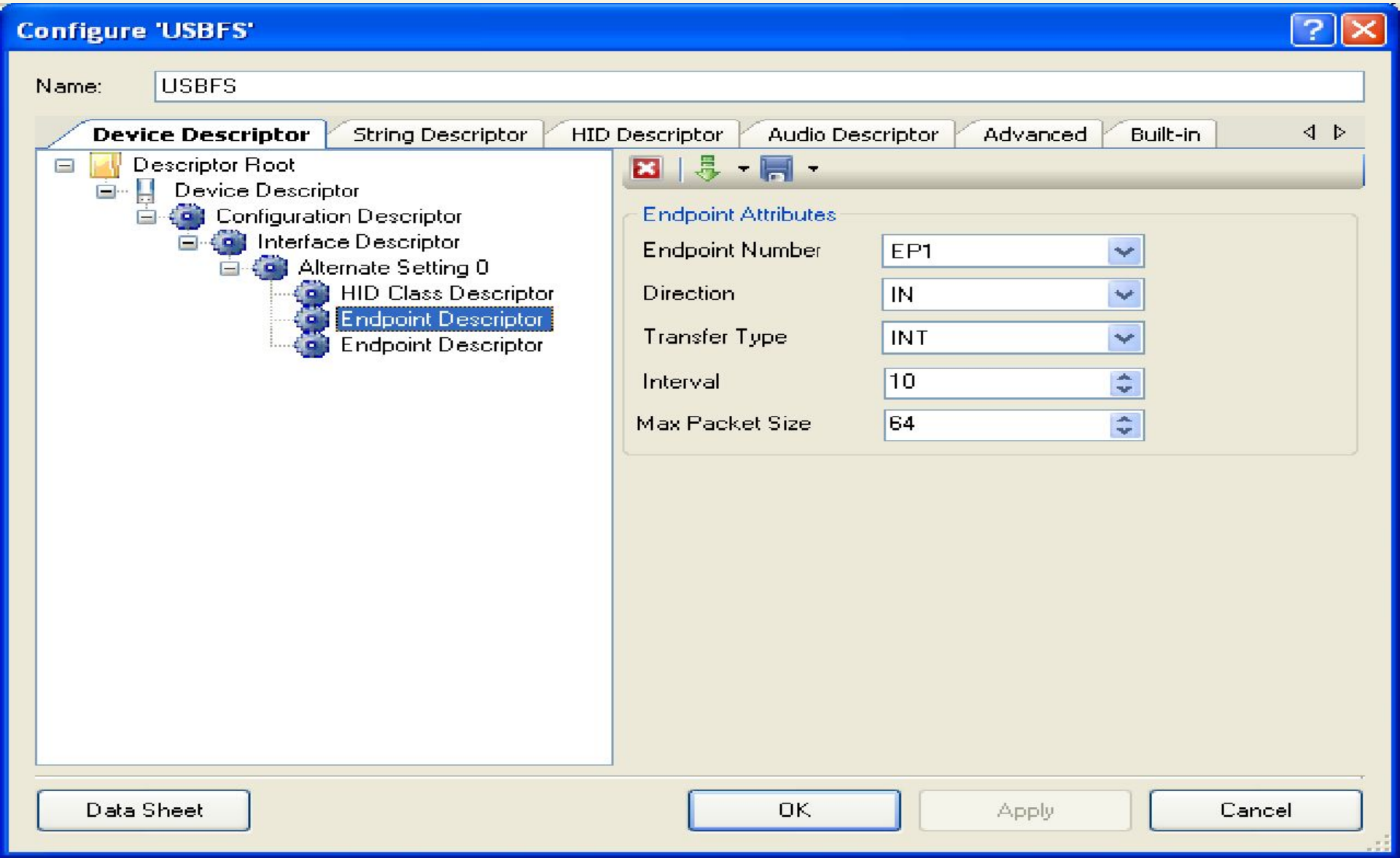
Interface Descriptor



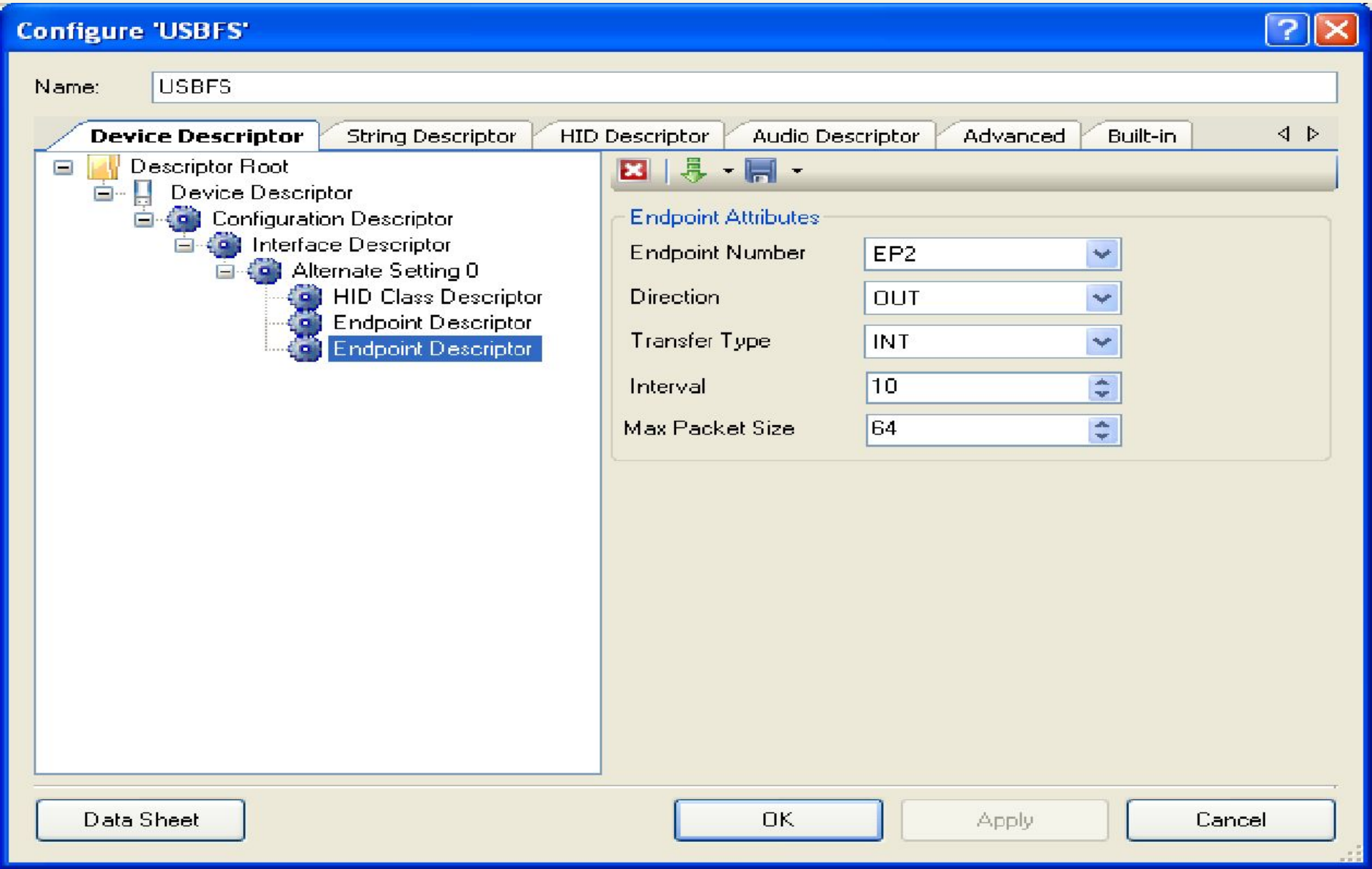
HID Descriptor



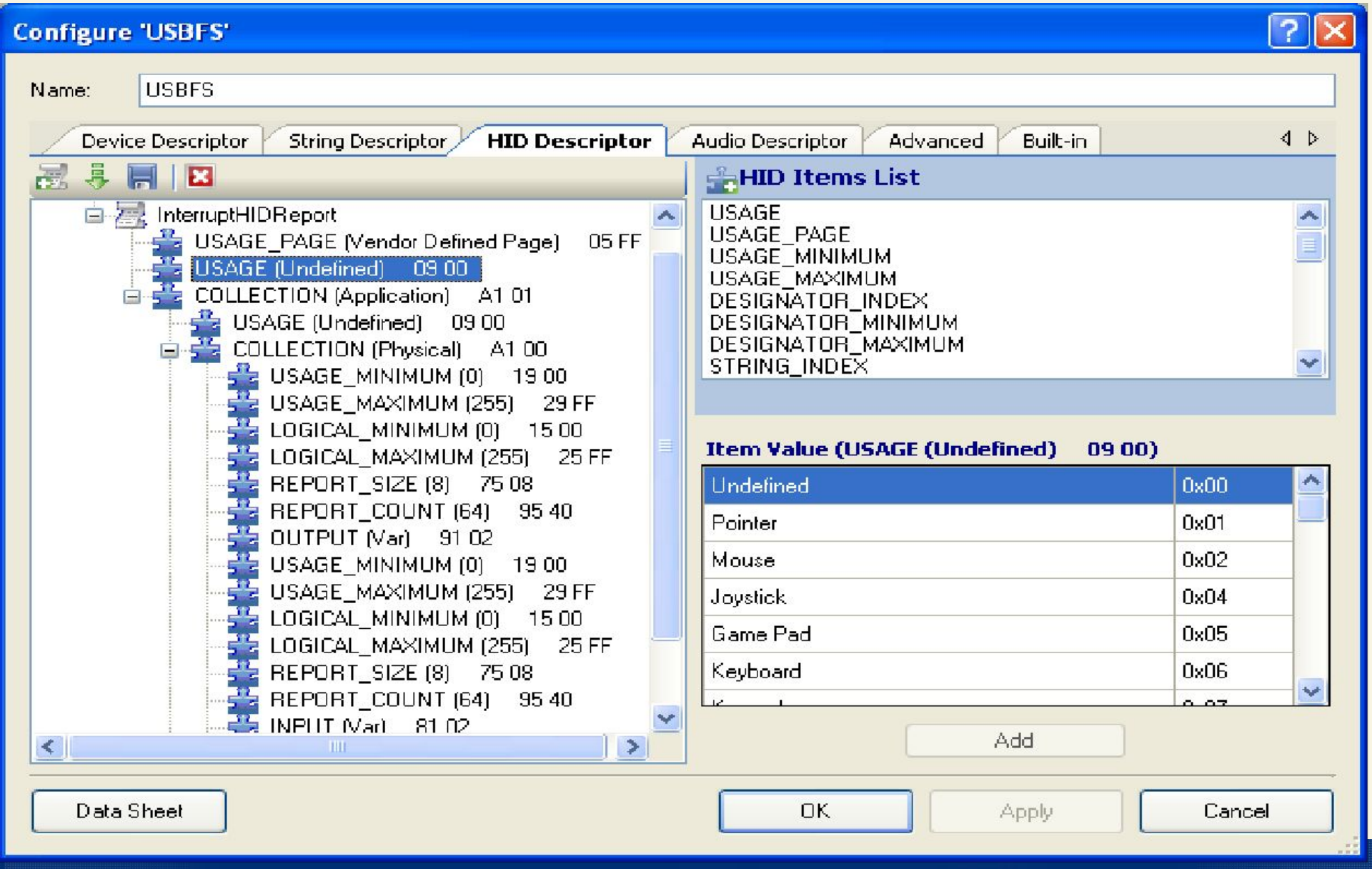
EndPoint Descriptor (in)



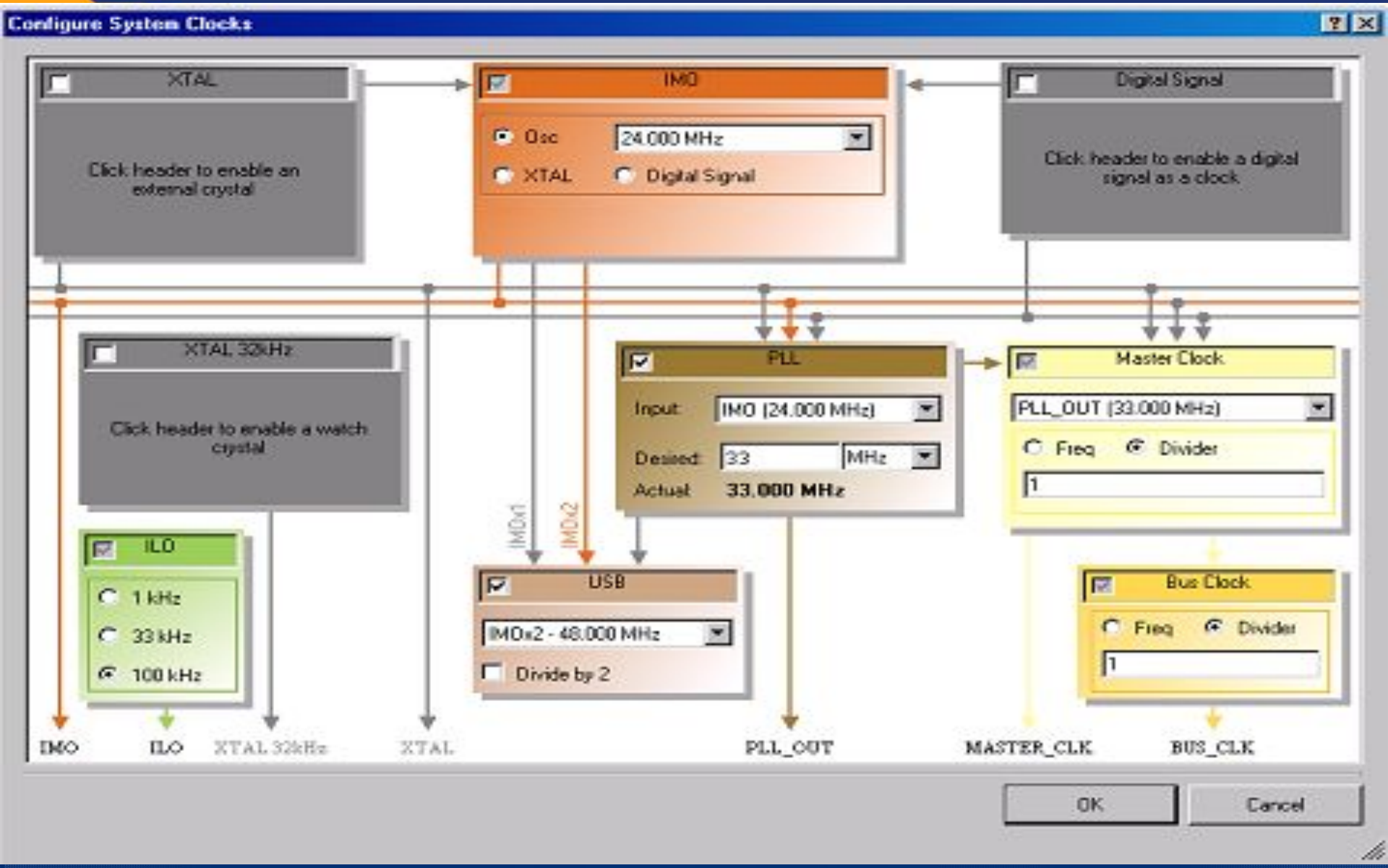
EndPoint Descriptor (out)



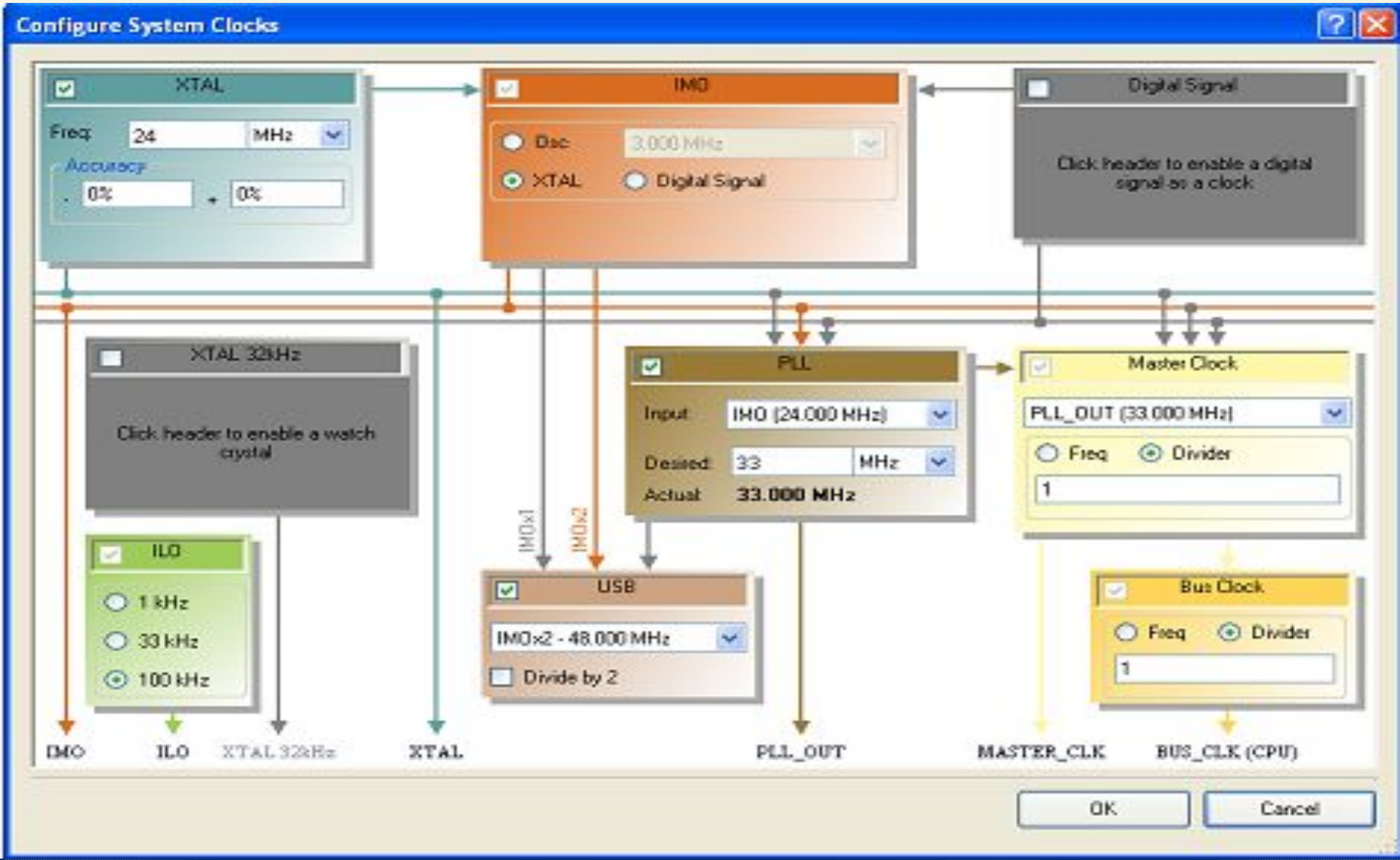
Descriptor Report



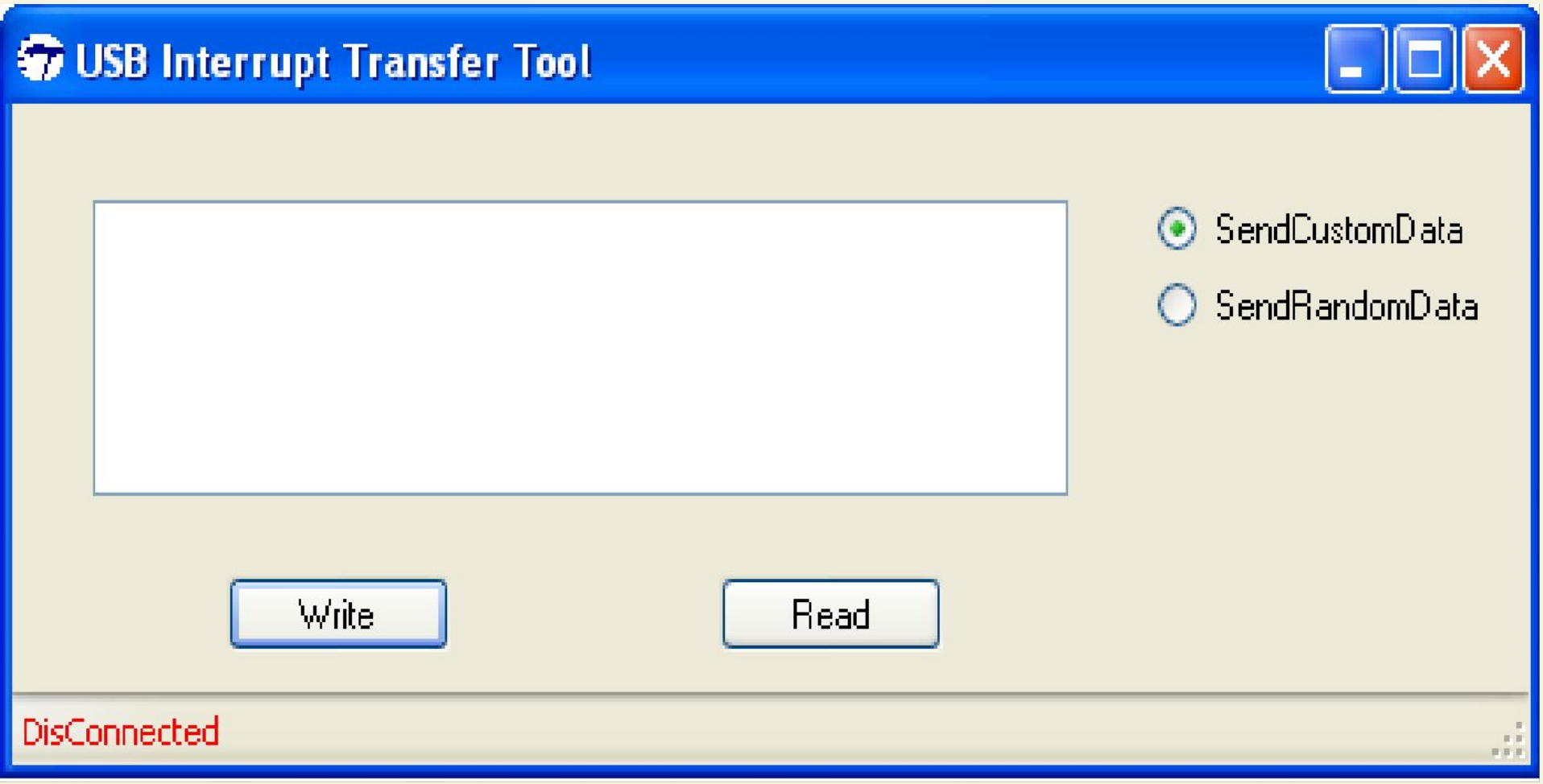
System Clock Configuration for PSoC 3



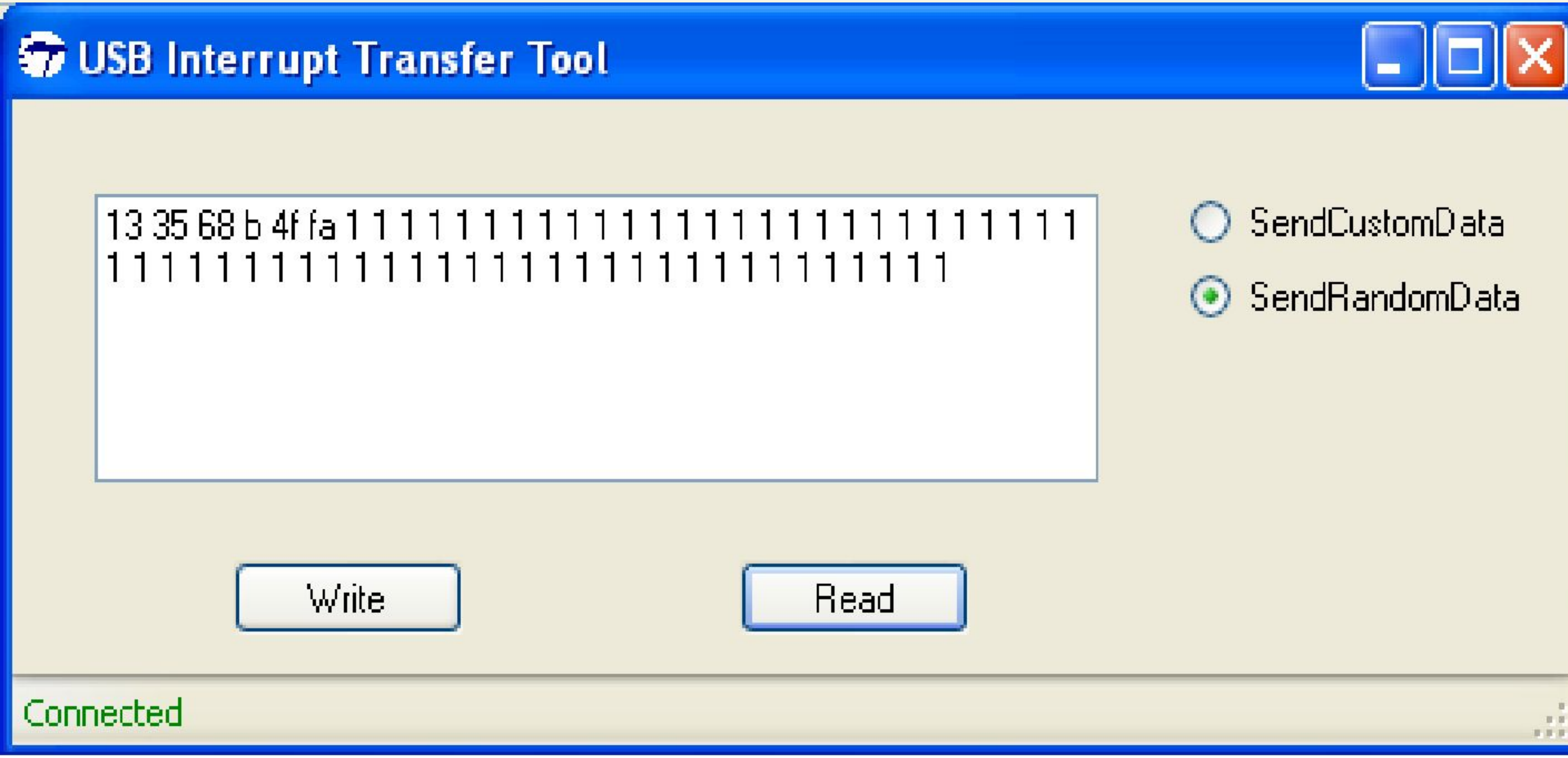
System Clock Configuration for PSoC 5



GUI project(C#)



GUI project(C#)



USBFS APIs

Function	Description
USBFS_Start()	Activates the component for use with the device and specific voltage mode.
USBFS_Init()	Initializes the component's hardware.

byte[] ReadFromPSoC();

WriteToPSoC(byte[] PSoCData);

FindFirstTouchDevice(int VendorId, int ProductId).

USBFS main.c_1

```
*  Project Name: Example_USB
*  Device Tested: CY8C3866AXI,
*  Software Version: PSoC Creator FCS 3.3
*  Complier tested: Keil(C51) and GCC
*  Related Hardware: CY8CKIT-030 or CY8CKIT-050
* Description:
•This project is aimed at transferring 64 or less bytes of data from the
•* PC to the PSoC 3/5 device  using the USB interrupt Out endpoint.
•* The data sent is modified by the PSoC 3/5 device and is sent back to
•* the PC using interrupt In endpoint.
•* A GUI is created for sending data from the PC to the  PSoC 3/5 device
•* and to read back the data from the PSoC 3/5 device.
* A full speed USB component in PSoC 3/5 is used in this project which
* enumerates as a generic HID class device
•The project needs the connection of the USB cable from the DVK to PC.
```

USBFS main.c_2

```
 * The USB uses P15[6] and P15[7].
•Once the component is placed on the TopDesign, these pins are
•*automatically blocked.
 * The data from the USB transceiver is then fed to the USB cable which
 * is connected to J9 in DVK
 */
#include <device.h>
#define TRUE 1
#define FALSE 0
/* Macro to store the Device Id */
#define DEVICE_ID 0
/* Macro to store the Out Endpoint number */
#define OUT_ENDPOINT 2
/* Macro to store the In Endpoint number */
#define IN_ENDPOINT 1
```

USBFS main.c_3

```
/* Macro to store the maximum number of bytes that can be received */
#define MAX_NUM_BYTES  64

/* Function to process data at the endpoints */
void ProcessEP2Data(void);

/* This variable is declarded in the USBFS_drv.c file.
 * This variable holds the configuration data for the endpoints */
extern T_USBFS_EP_CTL_BLOCK USBFS_EP[];

/* This variable is accesed inside the WSB Interrupt (see file USBDS_episr.c)
 * The variable is set to 1 when the interrupt occurs (The endpoint 2 ISR is named as USBFS_EP_2_I
 * and it is set to 0, once the data processing is completed in main.c */
uint8 USB_interruptFlag = 0;

void main()
{
    /* Enable the global interrupts */
    CYGlobalIntEnable;

    /* Start USBFS Operation for the DEVICE_ID and with 3.3V operation  */
    USBFS_Start(DEVICE_ID, USBFS_5V_OPERATION);
    LCD_Start();

    /* Wait for Device to enumerate. The function will return a 0 until the enumeration is complet
    while(FALSE == USBFS_bGetConfiguration())
    {
        /* Waiting for enumeration */
        ;
    }
```

USBFS main.c_4

```
97              ;
98          }
99
100         /* Once the device is enumerated, Enable Endpoint 2 for receiving data */
101         USBFS_EnableOutEP(OUT_ENDPOINT);
102
103         while(TRUE)
104         {
105             /* Program Loop */
106             /* Check if the Interrupt has occurred */
107             if(USB_interruptFlag == TRUE)
108             {
109                 /* Clear the Interrupt flag */
110                 USB_interruptFlag=0;
111
112                 ProcessEP2Data();
113             }
114         }
115     }
116
117 /*
118  * Summary:
119  *  This function copies the received data bytes from Out endpoint,
120  *  Increments the data and puts it back to the In endpoint buffer
121  * Parameters:
122  *  None
123  * Return
124  *  None
125  * Other details:
126  *  This function is called when the Out endpoint ISR occurs */
```

USBFS main.c_5

```
128  void ProcessEP2Data()
129  {
130      uint8 count, index;
131      uint8 USB_data[MAX_NUM_BYTES];
132
133      /* Get the number of data received */
134      count = USBFS_wGetEPCount(OUT_ENDPOINT);
135
136      /* Read the Out endpoint data into a user buffer (USB_data) */
137      USBFS_ReadOutEP(OUT_ENDPOINT, USB_data, count);
138
139      /* Enable Out Endpoint, to receive next set of data */
140      USBFS_EnableOutEP(OUT_ENDPOINT);
141
142      LCD_Position(0,0);
143      LCD_PrintHexUint8(USB_data);
144
145      /* All the received data is incremented by 1 by PSoC before sending back to the PC */
146      for(index = 0; index < count; index++)
147      {
148          /* Increment the received data */
149          USB_data[index]++;
150      }
151
152      /* Load the incremented data buffer back to in endpoint.
153      * This data will be received by the PC */
154      USBFS_LoadEP(IN_ENDPOINT, USB_data, count);
155
156  }
157  /* [] END OF FILE */
```

Overview:

Activate and use
the USB on the DVK board
and
output the results
to the LCD Character screen and Leds.

USBFS PSoC@3 v 5

Рекомендована література :

***Основна*:**

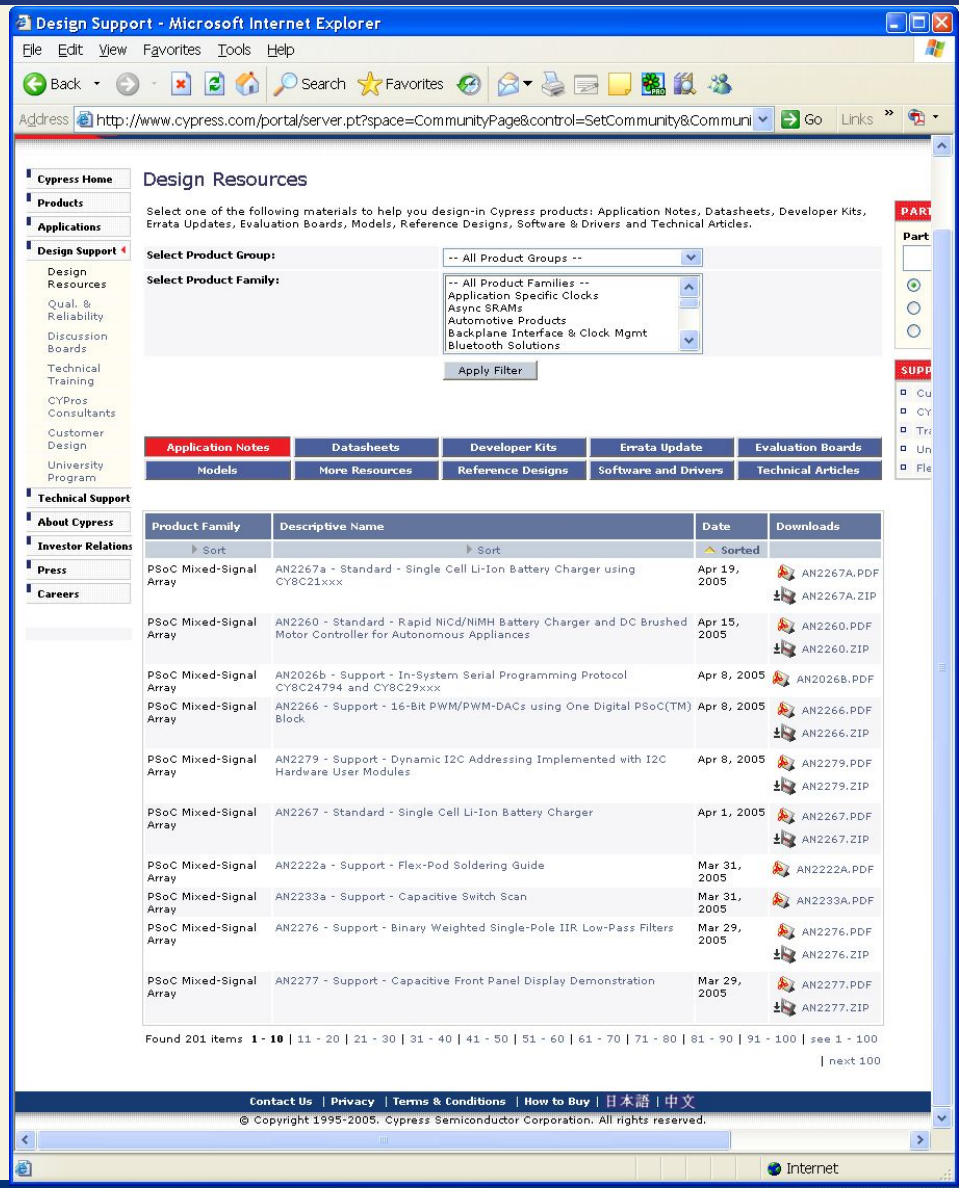
1. Гуров П. Практика програмирования USB. Санкт-Петербург «БВХ-Петербург» 2006 р. - 621 с.
2. PSoC® 3 Architecture TRM (Technical Reference Manual) – 379 с.

***Додаткова*:**

1. PSoC® Creator™ Component Datasheet. Full Speed USB (USBFS) – 66 с.

Приклади застосування МК PSOC

На сайті фірми Cypress знаходиться більше 200 Application Notes і Reference Designs, які ілюструють області застосування мікроконтролерів PSoC.



Мікропроцесорна техніка

(лекція 11, кінець)

Благітко Б.Я.

2019 р.