
電子工作専科
www.st.com/stm32f3discovery
ARDUINO
UNO

USB HID Device (PC側)

(2015.6.15 作成)
(2015.10.10 修正)
(2020.11.28 更新)

このページでは任意の64バイトまでのデータをUSB HIDプロトコルに基づいてPCとマイコン間で送受信するためのホスト側(PC側)について紹介したいと思います。マイコン(STM32)側はこちらで紹介しています。

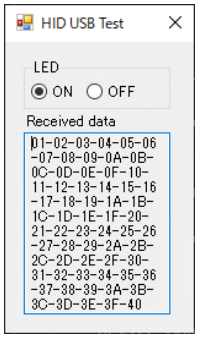
またここで言うホスト = PC = Windows = C#という環境での説明です。実際はAndroidやMacでも使えるとは思うのですが、単に管理人がWindows以外あまり触ったことがないというだけですのでそこはご容赦を。

まずHIDデバイスですが、基本的には名前(Human Interface Device)が表すようにキーボードやマウス用のプロトコルです。このプロトコルを利用して任意のデータを送受信できるようにしています。そのあたりの説明はこちら。

Windows(C#)でHIDデバイスを使用する例は検索すればいくつか出てきます。かつてはこのページでもこのライブラリを使用した方法を紹介していたのですが、長くメンテナンスされておらず、またライセンスも小難しいので新たなものに変更いたしました。

新たなプログラムはHidLibraryを使用したものでMITライセンスですし、nugetを使用して容易に導入することができます。

プロジェクト作成



とりたてて難しい設定などありません。プロジェクトを新規作成し、nugetからHidLibraryを検索してインストールするだけです。使い方は以下のサンプルコードを見てください。

HidLibraryはドキュメントもないので管理人もサンプルコードを見ながら作成しただけなので使いこなせているとは思いませんが、必要十分な動作はできていると思います。

一点だけ注意点があるとすればPCとマイコン間では64バイトのデータを送受信しますが、それとは別に最初に1バイトReport IDというものが付加されて実際は65バイトの通信を行っています。本当はマイコン側もしっかり作れば65バイトすべて使うことができるのですが、ここではReport IDを無視して64バイトだけ使用するようにしています。このため送信コード内では

```
rep = new HidLibrary.HidReport(65);
```

として65バイト確保しています。

サンプルコード

少し長いですが、サンプルコードを以下に載せます。

```
using System;
using System.Linq;
using System.Windows.Forms;

//https://csharpdoc.hotexamples.com/class/HidLibrary/HidDevice

namespace TestHID
{
        public partial class Form1 : Form
    {
        public Form1()
        {
```

```
        InitializeComponent();
    }

    const int VendorId = 0x483;     // STMicroelectronics
    const int ProductId = 0x5750;  //
    HidLibrary.HidDevice _device;

    private void Form1_Load(object sender, EventArgs e)
    {
        timer1.Interval = 500;
        timer1.Start();
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        TryDeviceConnect();
    }

    void TryDeviceConnect()
    {
        _device = HidLibrary.HidDevices.Enumerate(VendorId, ProductId).FirstOrDefault();

        if (_device != null)
        {
            _device.Inserted += () => Invoke((MethodInvoker)(()=>
            {
                groupBox1.Enabled = true;
            }));

            _device.Removed += () => Invoke((MethodInvoker)(() =>
            {
                groupBox1.Enabled = false;
                timer1.Start();
            }));

            _device.MonitorDeviceEvents = true;
                        _device.ReadReport(OnReport);
            _device.OpenDevice();
            panel1.Enabled = true;
            timer1.Stop();
        }
            }

            void OnReport(HidLibrary.HidReport report)
            {
                    this.Invoke((MethodInvoker)(() =>
                    {
                            textBox1.Text = Convert.ToString(BitConverter.ToString(report.Data));
                    }));
                    if (_device != null)
                            _device.ReadReport(OnReport);
            }

    private void radioButton_CheckedChanged(object sender, EventArgs e)
    {
        if (!((RadioButton)sender).Checked) return;  // prevent to be called twice
```

```
            HidLibrary.HidReport rep;
            rep = new HidLibrary.HidReport(65);
            int i;
            if (radioButton1.Checked)
            {
                for (i = 0; i < rep.Data.Length; i++)
                    rep.Data[i] = (byte)i;
            }
            else
            {
                for (i = 0; i < rep.Data.Length; i++)
                    rep.Data[i] = (byte)(rep.Data.Length - i);
            }
                        _device.WriteReport(rep);
        }
        }
}
```

接続先のVIP、PIDを固定しており、このIDのデバイスが接続されていなければタイマを使用して500ms毎に接続に挑戦しています。一度接続されれればデータの送受信が可能になります。
またデバイスが外されるとRemovedイベントが発生するので再度タイマを開始して再度接続されるのを待つようになります。

このプログラムとデバイス側のプログラムを合わせて使用することでPCとデバイスかでデータ送受信が出来ることが確認できると思います。

コメント: 0

名前: *

#1

コメント: *

プライバシーポリシー が適用されます

送信

＊ 入力必須