■光学工房

気軽にUSB

研究には，さまざまな機器を駆使して実験を行う必要に迫られる場合があります．さまざまな機器のさまざまなパラメーターを制御しながら計測するためには，それらの設定を手作業で操作を行うのではなく，機器をPC（personal computer）に繋げて実験を行います．一昔前なら，RS-232（Recommended Standard 232）やGP-IB（General Purpose Interface Bus, IEEE 488）での制御で事足りていました．しかし，USB（universal serial bus）がPCにほぼ標準的に搭載されるようになってから，USBをインターフェースにもつ機器が増えてきて，USBによる機器制御が必要になってきました．今回，USBをインターフェースにもつ分光器（Ocean Optics社：HR2000）からのスペクトル取得に迫られましたので，USBによるデータ取得プログラミングについて紹介したいと思います．なお，確認はしていませんが，同社のUSB2000とコードは互換性があります．また，少し拡張すれば簡単に最近のUSB分光器からのデータ取得も可能になると思います．

さて，Windows以前のプログラミングは非常に簡単でした．例えば，各ポートに対する命令さえわかれば，直接I/Oポートに命令を送ることで制御ができたからです．しかしながら，Windowsプログラムでは直接ハードウェアを制御することはできず，かならずデバイスドライバーを通して制御を行わなければなりません．デバイスドライバーの作成も可能ですが，敷居が高くそう安々とは行えません．もちろん，機器メーカーは，デバイスドライバーを用意し，これを使うためのライブラリーを提供していますが，これらが非常に高価である場合があります．そこで，メーカー提供のライブラリーを使わない，公開されている通信コマンドを用いたプログラミングを紹介します．

1. デバイスドライバー UUSBD

Windowsのプログラミングですので，USB機器を制御するためにはデバイスドライバーが必要となります．ここでは，柏野政弘氏が公開（http://www.otto.to/kasiwano/newpage17.htm）している汎用USBデバイスドライバーuusbd.sysとダイナミックリンクライブラリーuusbd.dllを用いて制御してみました．

UUSBDはユーザープログラムからのread/writeをそのままデバイスへのUSBの送受信コマンドとして通信します．したがって，ユーザープログラムはUUSBDを通じて直接USB機器とデータをやり取りするようにプログラミングできます．

さて，UUSBDでプログラミングする際に必要な情報は，

1．Vendor ID
2．Product ID
3．データ転送のためのパイプ番号
4．USBコマンド

です．これらの情報は，Ocean Optics社の場合ではホームページのEngineering docs（http://www.oceanoptics.com/technical/engineeringdocs.asp）で公開されています．

UUSBDのインストールに関しては，柏野政弘氏のホームページ等を参考にしてください．インストール後，uusbd.infファイルのVID（vendor ID）とPID（product ID）を3か所ずつ設定する必要があります．Ocean Optics社のVendor IDは2457，HR2000（USB2000）のProduct IDは100A（1002）なので，これらを書き換えます(いずれも16進数表記です)．なお，VID，PID等を複数記述することも可能です．

2. プログラミング

実際のプログラミングでは，USBデバイスのオープン（Uusbd_Open，またはUusbd_open_mask），

リスト 1

```
void SetIntegTime(int integration_time )
{
    HUSB husb;
    HANDLE com_out;         // handle for command output pipe
    BYTE data[3];
    DWORD  checknum;

    husb=Uusbd_Open();
    com_out=Uusbd_OpenPipe(husb, 0, 0);  // interface number 1, pipe 1

    data[0]=0x02;              // 0x02 is a command for integration time
    data[1]=(BYTE)(integration_time%0x100);
    data[2]=(BYTE)(integration_time/0x100);

    WriteFile( com_out, data, 3, &checknum, NULL);

    CloseHandle(com_out);
    Uusbd_Close(husb);
}
```

USB デバイスのパイプハンドルの取得（Uusbd_OpenPipe），データの送受信（ReadFile, WriteFile：いずれも Windows のファイル操作 API 関数），パイプのクローズ（CloseHandle：Windows の API 関数），デバイスのクローズ（Uusbd_Close）を順に行います．

USB は複数の機器を数珠つなぎで接続することができます．したがって，どの USB デバイスに通信を行うのか，指定する必要があります．Uusbd_Open 関数は uusbd.inf に書かれた VID と PID に合致する USB デバイスをオープンします．したがって，合致するものが複数ある場合は，そのうちの 1 つが選択されます（どのデバイスが選択されるかは，繋ぎかたに依存します）．もし PID 等で選択したい場合は，Uusbd_open_mask 関数を使って選択します．

USB はシリアル通信ですが，通信データを時分割して送受信し，USB デバイスの複数の FIFO メモリーと PC の複数のバッファー間でデータをやり取りすることができます．この USB デバイスの FIFO メモリーを，ハードウェアレベルではエンドポイントとよびます．ユーザープログラムでは，エンドポイントと直接やり取りするのではなく，パイプとよばれる仮想的な通信経路を使ってデータを送受信します．したがって，ユーザーとしては，何番パイプで，どのようなデータをやり取りすればよいかがわかれば，プログラミングが可能となります．HR2000 では，EP2（エンドポイント 2）がコマンドの送信とスペクトル情報の受信，EP7（エンドポイント 7）が分光器情報の受信のために用いられます．なお，HR2000 のデータシートにはエンドポイントとパイプの関係が示されていませんでしたが，他の分光器のデータシートからの類推と Try & Error から，パイプ 1 が EP2 の送信，パイプ 2 が EP7 の受信，パイプ 4 が EP2 の受信であることがわかりました．

分光器との通信は，パイプ 1 に命令（先頭バイトがコマンドを表し，その後に情報を付加する）を送ることで行います．リスト 1（以下，C 言語のソースの一部．簡単のためエラー処理はない）に，分光

リスト 2

```
void QueryInfo(char *buf, int com) // buf has to be larger than 16
{
    HUSB husb;
    HANDLE com_out;            // handle for command output pipe
    HANDLE com_in;             // handle for data input pipe
    BYTE odata[2], idata[18];
    DWORD  checknum;
    int i;

    husb=Uusbd_Open();
    com_out=Uusbd_OpenPipe(husb, 0, 0);  // interface number 1, pipe 1
    com_in =Uusbd_OpenPipe(husb, 0, 3);  // interface number 1, pipe 4

    odata[0]=0x05;             // 0x05 is a command for query information
    odata[1]=(BYTE)com;

    WriteFile(com_out, odata, 2, &checknum, NULL);
    ReadFile( com_in, idata, 18, &checknum, NULL);

    for(i=0; i<17-2; i++){      // data starts from 3rd byte
        buf[i]=(char)idata[i+2];
    }
    buf[i]='\0';

    CloseHandle(com_out);
    CloseHandle(com_in);
    Uusbd_Close(husb);
}
```

器の積算時間の設定関数を示します．積算時間の設定を表すコマンドが 0x02 で，これを先頭バイトとし，続いて積算時間を表す情報を付加して，パイプ 1 から送信しています．

分光器からデータを受け取るには，パイプ 4 を用います．リスト 2 は，分光器の各種設定情報を取得する関数です．まず，パイプ 1 の先頭バイトに分光器の設定情報の受信を表すコマンド 0x05 と，それに続いてどのような設定の返信を要求するか表すコマンドを加えて送信します．例えば，0x00 はシリアルナンバー返信要求を表します．そして，引き続いてパイプ 4 を通して情報を受信します．

同様に，スペクトルも取得することができます．このときのコマンドは 0x09 で，パイプ 1 でこれを送信します．スペクトルの取得はパイプ 2 を通じて行います．HR2000 の場合，1 つのスペクトルは 16 ビット 2048 個のデータからなりますが，各データは上位ビット，下位ビットに分けて，さらに 64 個ずつに分割されて送られてきます．なお最後にデータが終わったことを表す，0x69 の値が 1 バイト付加されています（リスト 3）．

以上のように，汎用デバイスドライバー UUSBD を用いると，簡単に USB 機器を制御することができます．もちろん，機器の USB コマンドが公開されている場合に限られますが．

さて，筆者のグループでは，Igor Pro（wavemetrics 社，http://www.wavemetrics.com）というデータ処理ソフトウェアを，機器制御にも用いています．今回，HR2000 と通信するプラグイン（Igor

リスト 3

```
void GetSpectrum(int *wave)     // wave has to be larger than 2048
{
    HUSB husb;
    HANDLE com_out;            // handle for command output pipe
    HANDLE data_in;            // handle for spectrum input pipe
    BYTE   idata1[64], idata2[64];
    BYTE   odata, idata;
    DWORD  checknum;
    int i, j;

    husb=Uusbd_Open();
    com_out=Uusbd_OpenPipe(husb, 0, 0);  // interface number 1, pipe 1
    data_in=Uusbd_OpenPipe(husb, 0, 1);  // interface number 1, pipe 2

    odata=0x09;                // 0x09 is a command for getting a spectrum
    WriteFile( com_out, &odata, 1, &checknum, NULL);

    for(i=0; i<32; i++){
        ReadFile( data_in, idata1, 64, &checknum, NULL);
        ReadFile( data_in, idata2, 64, &checknum, NULL);
        for(j=0; j<64;j++){
            wave[i*64+j]=idata2[j]*0x100+idata1[j];
        }
    }
    ReadFile( data_in, &idata, 1, &checknum, NULL);

    CloseHandle(com_out);
    CloseHandle(data_in);
    Uusbd_Close(husb);
}
```

Pro では XOP とよばれています）も作成しました．使用してみたい方，ソースファイルが必要な方は，筆者（mamoru@me.es.osaka-u.ac.jp）まで直接お問い合わせください．

なお，例に掲げたプログラムの一部，ならびに XOP は，ユーザーの自己責任の範囲内で使用してください．これらの使用によって生じた不具合や不利益などについて，その妥当性を問わず日本光学会ならびに著者は一切責任を負いませんので，ご了承ください．

（大阪大学　橋本　守）