USBのI/O性能と改善のためのヒント

内藤 竜治
Ryuji Naitou

前章では，パソコンからUSBを経由してLEDを点灯させたり，スイッチの入力を読み取ったりする方法について紹介しました．

ところが，この方法を応用して複雑なシステムを制御しようとすると，動作速度が遅くて使いものにならないのではないかと思います．

本章では，USBを通じた機器操作のパフォーマンスを向上させるためのヒントを紹介します．

付録基板のパフォーマンスを知る

パフォーマンスの向上は，現状を知ることから始めましょう．

まず，通信速度を測ることにします．USBでは一般に，IN(ターゲット→ホスト)とOUT(ホスト→ターゲット)の速度が異なるので，両方測ることが必要です．

● IN(ターゲット→ホスト)側の通信速度を測る

IN側の速度の測り方はとても簡単です．1Mバイト程度のデータを転送したときの通信時間を測り，

通信速度＝データ・サイズ÷通信時間

で求めます．

256バイトを4096回送信するマイコン側のプログラムは次のようになります．

```
for(i=0; i<4096; i++) {
  trg_senddata(tmp,256);
}
```

このルーチンを呼び出す前に，tmpというバッファにはあらかじめ適当な文字列をセットしておきます．

実際に測定してみると23秒かかったので，通信速度は46Kバイト/sということになります．この値を10倍すればRS-232-Cに換算したビット・レートになります．

RS-232-Cに換算すれば455 kbpsなので，まずまずの速度でしょう．速度は使用しているパソコンの処理速度にも若干影響されるので，あくまでも目安と考えてください．

リスト1　パラレル・ポートの反応にかかる時間を測るためのプログラム
デバイス・ドライバgiveio.sysを使う

```
void pptw2k_outp(unsigned short addr,unsigned char data){
          // I/Oポートを直接操作するためのルーチン (Borland C++用)
          _EDX = addr;
          _AL  = data;
          __emit__(0xEE); // アセンブラのout命令を直接書く
}

int main()
{
          ・・・(略 ここでgiveioをオープンする)・・・
          for(int i=0;i<1000;i++) {
                    pptw2k_outp(0x378, i & 1);
          }
}
```

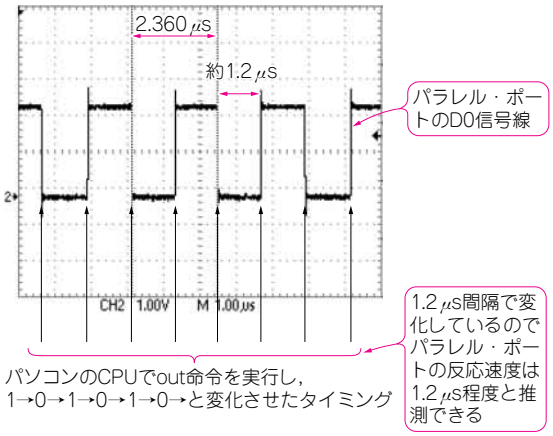
※このプログラムを実行するにはgiveio.sysが必要．giveio.sysは特権命令である「out」や「in」命令をユーザ・アプリケーションから自由に使えるようにしてしまうという恐ろしいデバイス・ドライバ．下記から入手できる．
ftp://ftp.nsk.su/.3/magazins/ddj/1996/1996.05/directio.zip

リスト2　シリアル・ポートが反応する時間を測るためのプログラム
WindowsAPIのWriteFile関数を使っているためパラレル・ポートより反応速度は遅い

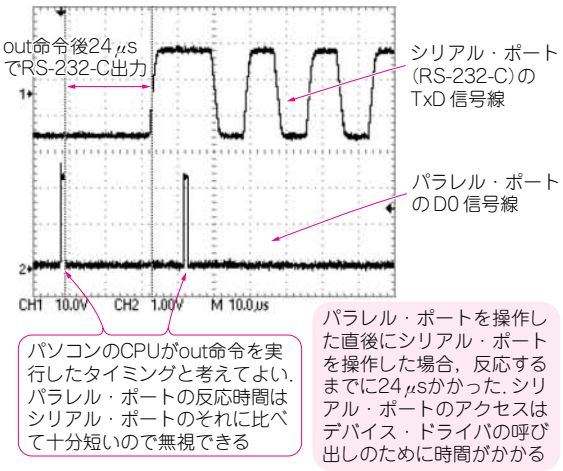
```
DWORD rbytes;
buf[0] = 0xaa;

pptw2k_outp(0x378,1);
pptw2k_outp(0x378,0);
WriteFile(hCom,buf,1,&rbytes,NULL); // データ送信
pptw2k_outp(0x378,1);
pptw2k_outp(0x378,0);
```

第2章Appendixの「簡単なCOMポート入出力プログラム(Simple Term)」のコードを使ってCOMポートをオープンし，DCB(デバイス制御ブロック)をセットしておくこと．レガシーI/Oの通信速度は115200bpsにセットしてある．



（a）パラレル・ポート



（b）シリアル・ポート

図1　レガシーI/Oの反応にかかる時間
パラレル・ポートは1.2 μsで反応し，シリアル・ポートは24 μsかかる

● OUT(ホスト→ターゲット)側の通信速度を測る

次はOUT側の通信速度を測ります．付録基板ではなぜかOUT側の方が遅いので，転送サイズを少なくして測ることにします．

256バイトを256回受信するマイコン側のプログラムは次のようになります．

```
for(i=0;i<256;i++) {
  trg_recvdata(tmp,256);
}
```

65536バイト送信するのに要した時間は13秒だったので，OUT方向は5Kバイト/s程度ということになります．RS-232-C換算では50 kbpsとやや遅めです．

USB2.0フル・スピードの通信速度は12 Mbpsと言われているので，ずいぶんと遅いように感じるかもしれませんが，アプリケーション・レベルではこの程度です．仮想COMポートは，もともとレガシーなシリアル・ポートをエミュレートするものなので，この程度出ていれば十分です．

● OUT方向の通信速度が遅い理由とその対策

USBの送信も受信も，ユーザ・プログラム内のバッファとUSBファームウェア内のバッファとの間でデータを1バイトずつコピーしていくことには変わりありませんが，通信速度に差が生じる理由は，データ・コピーに伴うチェックの頻度にあります．

IN方向の通信（78K0マイコンが送信）はsvUSB_datawriteという関数で行いますが，この関数は最大64バイトをまとめて送信できるようになっています．

OUT方向の通信（78K0マイコンが受信）はsvUSB_bytereadという関数で行いますが，この関数は1バイト単位で受信するようになっています．

したがって，受信では1バイトをコピーするたびに，バッファ内のデータ数や各種フラグをチェックしたり，USBのパケット・リードを行ったりするため，速度が低下していると推測されます．

高速なマイコンではないので，バッファの境界チェックなどの処理がダイレクトにパフォーマンスに影響しています．

バッファの境界をチェックする処理は，トラ技BIOSのソース・コードBIOScore.cの中にあるusbRecvDataという関数の，

```
while(1)
{
  if(!usb_connected()) return;
  if(svUSB_getrecvcount() != 0)
  break;
}
```

という部分です．このチェックをfor文の外に出すと，速度を100 kbps程度まで向上できます．

ただし，この高速化を行うと通信中にUSBポートがクローズされたときの動作が不安定になります．トラ技BIOSでは，遅くてもよいので確実に受信ができるようなプログラムにしています．

より高速化する必要のあるアプリケーションを作成する場合は，これらのチェックを安全に減らすような工夫をするとよいでしょう．

● 通信速度が速ければ良いというわけではない

付録基板のUSBの通信速度は，レガシーのシリアル・ポートとあまり変わらないということが分かりま