GL シリーズ･データ受信時の注意点

弊社データロガーGL シリーズにて、弊社提供のサンプルプログラム（以降 GLSample）を参考とした場合で、測定データや収録データを通信で取得する際の方法、及び注意点を説明します。
尚、GLSample は VC++版と VB 版がありますが、本ドキュメントでは VC++版での説明となります。VB 版をご使用の方は大変恐縮ですが、関数名、ファイル名等は VB 版へ読み替えて頂きますよう、お願い申し上げます。（注意点、考え方は VC++版も VB 版も同じです。）

１．テキスト形式について

通常、弊社 GL シリーズの I/F コマンドの通信はテキスト形式となります。
テキスト形式は、文字列を送受信する為の形式で、文字列の最後が改行コード(CR/LF)、もしくは NULL 文字で終端されます。
それぞれ文字コードは、CR=0x0d、LF=0x0a、NULL 文字=0x00 の値となります。

弊社 GLSample では CDevIo::ReadLn()関数にて文字列の受信を行っております。
当関数では、文字コード 0x0a,0x00 のどちらかが受信されるまで、1 文字ずつ受信を行い、受信文字列を指定されたバッファへ転送する関数となります。
詳細動作は CDevIo::ReadLn()関数のプログラムを参照ください。

２．バイナリ形式について

一部のクエリコマンドのレスポンスがバイナリ形式の場合があります。
特に、データの受信に関わる
:TRANS:OUTP:DATA?
:MEAS:OUTP:ACK?
:MEAS:OUTP:ONE?
上記３種類のクエリコマンドは、レスポンスがバイナリ形式となります。

バイナリ形式は、テキスト以外のデータを受信する場合に使用します。
この場合、テキスト形式で特別な意味を持つ文字コード（CR/LF/NULL）が別の意味として受信されます。

テキスト形式の場合、LF/NULL コードが受信されるまで受信を続けますが、バイナリ形式の場合は、受信するバイト数を指定して受信します。受信データは単なる数値の羅列として扱われ、データの並び方や、データの意味の解釈等は、上位の関数が適宜行います。

弊社 GLSample では CDevIo::ReadBinary()関数にてバイナリデータの受信を行っております。
関数の引数に受信するバイト数を指定し、指定されたバイト数のデータを指定されたバッファへ転送する関数となります。

３．CDevIo::SendQuery()関数について

CDevIo::SendQuery()関数は、内部で CDevIo::WriteLn()関数でコマンド文字列を送信し、CDevIo::ReadLn()関数でレスポンス文字列を受信しています。

従いまして、レスポンスがバイナリ形式のコマンドに CDevIo::SendQuery()関数を使用した場合、バイナリデータ内に CR/LF/NULL コードが含まれますと、正常に受信する事が出来ません。
現象として、想定するバイト数の受信前に受信が終了する、受信データ値がおかしい等が発生します。

４．バイナリ形式のデータ受信について

レスポンスがバイナリ形式の場合は、以下の様にして受信を行ってください。

①コマンドの送信は CDevIo::SendCommand()関数を使用してください。
②データの受信には CDevIo::ReadBinary()関数を使用してください。

(1) 最初に#6xxxxxx の形式で送信されるバイナリデータのバイト数が送信されます。このバイト数を読み込む為に、先ずは 8 バイトのバイナリ受信を行って頂き、その後に受信すべきバイト数を割り出してください。
(2) (1)で受信したバイナリデータのバイト数分のバイナリデータを順次受信してください。また、受信バイト数が大きい場合は、一度に受信せず、複数回に分けて受信される事を推奨します。

上記は GLSample の GetRecData()関数(GLSampleView.cpp 内(VB 版は StopProc.vb 内))に ":TRANS:OUTP:DATA?"コマンドを例に説明しています。(次ページに VC++版を抜粋)

【前略】

```
// データ取得コマンドの送信
if (pDev->SendCommand(":TRANS:OUTP:DATA?")) {
          // エラー
          goto CommErrExit;
}

// 受信サイズの受信
dwSize = 8;          // 8 バイト #6xxxxxx
if (pDev->ReadBinary(nBuf, dwSize, R_TIMOUT)) {
          // エラー
          goto CommErrExit;
}
if (dwSize != 8) {
          // エラー
          goto CommErrExit;
}

// 受信サイズを取り出す
nBuf[8] = 0;
if (nBuf[0] != '#') {
          // おかしい？
          // 予期せぬデータ
          // 余分な受信データが有るかも知れないが、とりあえず無視。
          // 念のため、本体の電源を一度切った方が良い。
          pView->AddLog("¥r¥n データを受信できませんでした。¥r¥n");
          goto CommErrExit;
}

dwRcvDataSize = atoi(&nBuf[2]);

if (dwRcvDataSize == 0) {
          // 受信データなし？
          break;
}

// 一時受信領域を確保
pBuf = new char[dwRcvDataSize + 4];       // 受信領域
                                          // ステータス(2byte)＋データ本体＋チェックサム(2byte)
if (pBuf == NULL) {
          // メモリが確保できなかった？
          goto MemErrExit;
}

// データの受信
dwSize = dwRcvDataSize + 4;     // ステータスとチェックサム領域を足す
if (pDev->ReadBinary(pBuf, dwSize, R_TIMOUT)) {
          // エラー
          goto CommErrExit;
}
if (dwSize != dwRcvDataSize + 4) {
          // エラー
          goto CommErrExit;
}
```

【後略】

以　上