```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using Microsoft.VisualBasic.ApplicationServices;

namespace WindowsFormsApplication1
{
    //メインフォーム
    public partial class frmMain : Form
    {
        //================================================================
        //ログデータ関連
        //================================================================
        static public int TXT_header_sectors = 3;   //TXT領域のセクタ数
        int LOG_Version = 6;                        //ログのバージョン
        int LOG_RecordBytes = 17;                   //ログの1レコードのバイト数
        public const int max_log_data_counts = 100000;      //10万行分のデータ★
        public struct LogData{
            public int      mode;           //mode
            public int      v, vt;          //速度、目標速度
            public int      angle, angle_t; //ハンドル角、目標角度
            public int      power;          //モータ出力
            public int      sv_pow;         //サーボモータの出力
            public int      fl,fr,rl,rr;    //各輪のモータ出力
            public int      slope_mode;     //slope_mode;
            public int      slope_sw;       //坂SWの状態
            public int      slope_cnt;      //出発してからの坂の数
            public int      trip;           //トリップメータ
            public float    batt;           //バッテリ電圧
            public int      gyro;           //ジャイロ出力値
            public int      side;           //サイドセンサの状態
            public int      pos_sens;       //ポジションセンサ
            public int      time;           //時間[ms]
            public StringBuilder    sens;   //センサの状態 (サイドセンサ含む)
            public int      pre_sens;       //先読みセンサ
        }

        static public LogData[] log = new LogData[max_log_data_counts]; //ログ
        static public int log_count;        //
        public string path;                 //
        StringBuilder str;
        int LogFileSize;                    //ログファイルの実質のサイズ
        //bool DriveIsFixedDisk = false;    //ハードディスクなら自動保存する。

        public frmOption frmOption1 = new frmOption();
        static public int SCROLLBAR_WIDTH = 20;
//      System.Threading.Mutex mut;         //多重起動禁止用のMutexオブジェクト

        //================================================================
        //グラフ関連
        //================================================================
        public const int graph_points = 12;
        public struct myGraphPoints{
            public bool enabled;
            public Single y, y1;
            public Pen pen;
            public Single scale, max, min;
        }
        static public myGraphPoints[] gp = new myGraphPoints[graph_points];

        static public int y0;                           //X軸
        static public int cur_n1=0, cur_x=0, cur_x1=0;  //グラフ上の現在,前の位置
        static public bool cur_show = false;            //カーソル表示
        static public Single graph_v;                   //グラフのX増分
        static public int scale_mode=1;                 //縮尺のモード
        static public Point scrPoint1, scrPoint2;       //グラフのスクロール座標

        //================================================================
        //バイナリファイルの圧縮保存
        //================================================================
        public void FileSave()
        {
            string path_save = path.Substring(0, path.Length - 4) + "_new.LOG";

            FileStream fsr = new FileStream(path, FileMode.Open, FileAccess.Read);
            int fileSize = (int)fsr.Length;         // ファイルのサイズ
            byte[] buf = new byte[fileSize];        // データ格納用配列
            fsr.Read(buf, 0, fileSize);
            fsr.Close();
            fsr.Dispose();

            FileStream fsw = new FileStream(path, FileMode.Create, FileAccess.Write);
            fsw.Write(buf, 0, LogFileSize);
            fsw.Close();
            fsw.Dispose();
        }

        //================================================================
        //ファイルを開く
        //================================================================
        public void FileOpen(string filename)
        {
            int WorkAddress, BuffAddress;
            int n=0, i;

            //拡張子のチェック
            if(filename.Substring(filename.Length - 4).ToUpper() != ".LOG"){
                MessageBox.Show("MCR用のLOGファイルではありません！");
                return;
            }

            path = filename;
            txtPath.Text = filename;

            //
            // テキストログの読み込み
            //
            System.IO.StreamReader TextFile;
            TextFile = new System.IO.StreamReader(path, System.Text.Encoding.Default);

            int c;
            c = TextFile.Read();
            txtHead.Clear();

            if(c == '#'){
                string Line;

                //ログのバージョンを読み込む
                Line = TextFile.ReadLine();
                LOG_Version = int.Parse(Line);
                //Form1.Text = Form1.Text + " (" + Line + ")";

                //1レコードのバイト数を決定
                if(LOG_Version <= 3){
                    LOG_RecordBytes = 12;
                }
                else{
                    Line = TextFile.ReadLine();
                    LOG_RecordBytes = int.Parse(Line);
                }

                //テキストログの読み込み
                n = 0;
                do{
                    Line = TextFile.ReadLine();
                    if(Line == "<END>") break;
                    if(n++ > 128) break;
                    txtHead.Text += Line + System.Environment.NewLine;
                }while(Line != null);
            }
            else{
                //LOG_Version = 6;
                //LOG_RecordBytes = 17;
            }
            txtHead.Text += String.Format("Log_Version = {0,3:d3}", LOG_Version) + System.Environment.NewLine;

            TextFile.Close();
            TextFile.Dispose();

            //
            // バイナリログデータの読み込み
            //
            FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read);
            int fileSize = (int)fs.Length;      // ファイルのサイズ
            byte[] buf = new byte[fileSize];    // データ格納用配列
            int readSize;                       // Readメソッドで読み込んだバイト数
            int ErrorCount = 0;                 // エラーの数

            //パラメータ
            int     time;       // スタートからの経過時間   [ms]
            sbyte   mode,       // モード
                    v,          // 現在の速度               [x10 m/s]
                    vt,         // 目標速度                 [x10 m/s]
                    angle,      // 現在のハンドル角         [°]
                    angle_t,    // 目標ハンドル角           [°]
                    power,      // 駆動モータの出力         [%]
                    slope,      // 坂の状態                 [slope_mode, sw, slope_count]

                    gyro,       // ジャイロセンサ出力
                    sv_pow,     // サーボモータの出力       [%]
                    fl,         // 前左モータ出力           [%]
                    fr,         // 前右モータ出力           [%]
                    rl,         // 後左モータ出力           [%]
                    rr,         // 後右モータ出力           [%]
                    side,       // サイドセンサ状態
                    pos_sens,   // ポジションセンサ状態
                    pre_sens;   // 先読みセンサ
            byte    sens,       // デジタルセンサ状態
                    batt;       // バッテリー電圧
            int     trip;       // スタートからの走行距離   [cm]

            if(c == '#') WorkAddress = TXT_header_sectors * 512;
            else         WorkAddress = 0;
            BuffAddress = 0;

            n = 0;
            fs.Seek(TXT_header_sectors * 512, SeekOrigin.Begin);

            lstView.Hide();
            lstView.Items.Clear();
            readSize = fs.Read(buf, WorkAddress, 512);

            time = 0;
            //*******************************************************
            //ログバージョン003以下の読み込み　ここから
            //*******************************************************
            if(LOG_Version <= 3){
                lblHead1.Text = "  time mode  sens   hnd ang  pow  vt   v slc    trip  diff   Batt   Gyro  ";
                lblHead2.Text = "                      A   B    C    D   E     F      G     ";

                while (WorkAddress < fileSize - 512){
                    mode    =   (sbyte)buf[WorkAddress + BuffAddress + 0];
                    sens    =   buf[WorkAddress + BuffAddress + 1];
                    v       =   (sbyte)buf[WorkAddress + BuffAddress + 2];
                    vt      =   (sbyte)buf[WorkAddress + BuffAddress + 3];
                    angle_t =   (sbyte)buf[WorkAddress + BuffAddress + 4];
                    angle   =   (sbyte)buf[WorkAddress + BuffAddress + 5];
                    power   =   (sbyte)buf[WorkAddress + BuffAddress + 6];
                    slope   =   (sbyte)buf[WorkAddress + BuffAddress + 7];
                    trip    =   buf[WorkAddress + BuffAddress + 8];
                    trip    <<= 8;
                    trip    +=  buf[WorkAddress + BuffAddress + 9];
                    batt    =   buf[WorkAddress + BuffAddress + 10];
                    gyro    =   (sbyte)buf[WorkAddress + BuffAddress + 11];

                    ErrorCount = (int)sens;

                    log[n].mode         = mode;
                    log[n].v            = v & 0x3f;
                    log[n].vt           = vt;
                    log[n].angle        = angle;
                    log[n].angle_t      = angle_t;
                    log[n].power        = power;
                    log[n].slope_mode   = (slope >> 6) & 0x03;
                    log[n].slope_sw     = (slope >> 4) & 0x03;
                    log[n].slope_cnt    = slope & 0x0f;
                    log[n].trip         = trip;
                //  log[n].batt         = batt;
                    log[n].batt         = ((float)batt)*8.0F * (float)frmOption1.nudBatt_a.Value + (float)frmOption1.nudBatt_b.Value;
                    log[n].gyro         = gyro;

                    //log_count         = n;

                    if((v & 0x80) != 0) log[n].sens = new StringBuilder("S");
                    else                log[n].sens = new StringBuilder(" ");
                    for(i=0; i<8; i++){
                        if(i != 4){
                            if((sens & 0x80) != 0) log[n].sens.Append("*");
                            else                   log[n].sens.Append("-");
                        }
```

```
                    sens <<= 1;
                }
                if((v & 0x40) != 0) log[n].sens.Append("S");
                else                log[n].sens.Append(" ");

                str  = new StringBuilder(String.Format("{0, 6}", time));
                time += 4;
                str.Append(String.Format("{0, 4}", log[n].mode));
                str.Append(log[n].sens);
                str.Append(String.Format("{0, 4}", log[n].angle_t));
                str.Append(" ");
                str.Append(String.Format("{0, 3}", log[n].angle));
                str.Append(" ");
                str.Append(String.Format("{0, 4}", log[n].power));
                str.Append(String.Format("{0, 4}", log[n].vt));
                str.Append(String.Format("{0, 4}", log[n].v));
                str.Append(" ");
                str.Append(String.Format("{0, 1}", log[n].slope_mode));
                str.Append(String.Format("{0, 1}", log[n].slope_sw));
                str.Append(String.Format("{0, 1}", log[n].slope_cnt));
                str.Append(String.Format("{0, 8}", log[n].trip));
                str.Append(String.Format("{0, 6}", log[n].trip));
                str.Append(String.Format("{0, 7:f1}", log[n].batt));
                str.Append(String.Format("{0, 7}", log[n].gyro));

                if (mode == -2)             //次のセクタへ
                {
                    int ii;
                    WorkAddress += 512;
                    readSize = fs.Read(buf, WorkAddress, 512);
                    BuffAddress = 0;

                    time -= 4;

                    for(ii=0; ii<ErrorCount; ii++){     //エラーの時はその分空行
挿入

                        log[n].mode         = 0;
                        log[n].v            = 0;
                        log[n].vt           = 0;
                        log[n].angle        = 0;
                        log[n].angle_t      = 0;
                        log[n].power        = 0;
                        log[n].slope_mode   = 0;
                        log[n].slope_sw     = 0;
                        log[n].slope_cnt    = 0;
                        log[n].trip         = 0;
                        log[n].batt         = 0;
                        log[n].gyro         = 0;

                        lstView.Items.Add("Err");
                        n++; if (n > 10000) break;
                    }
                }
                else
                {
                    BuffAddress += 12;
                    lstView.Items.Add(str);
                    n++; if (n > 10000) break;
                }

                if (mode == 0) break;       //modeが0なら終了
            }
        }
        //*********************************************************
        //ログバージョン003以下の読み込み終了　ここまで
        //*********************************************************


        //*********************************************************
        //ログバージョン004以上の読み込み　ここから
        //*********************************************************
        else if(LOG_Version >= 4){
            lblHead2.Text = "                                   A   B    C     D   E
 F   G   H   I      J        K     L    ";
            lblHead1.Text = "  time mode     sens    pos  hnd ang  sv    vt  v
fl  fr  rl  rr     x  slc   Batt  Gyro  ";

            while (WorkAddress < fileSize - 512){
                mode    =   (sbyte)buf[WorkAddress + BuffAddress + 0];
                sens    =   buf[WorkAddress + BuffAddress + 1];

                angle_t =   (sbyte)buf[WorkAddress + BuffAddress + 2];
                angle   =   (sbyte)buf[WorkAddress + BuffAddress + 3];

                sv_pow  =   (sbyte)buf[WorkAddress + BuffAddress + 4];

                vt      =   (sbyte)buf[WorkAddress + BuffAddress + 5];
                v       =   (sbyte)buf[WorkAddress + BuffAddress + 6];
                fl      =   (sbyte)buf[WorkAddress + BuffAddress + 7];
                fr      =   (sbyte)buf[WorkAddress + BuffAddress + 8];
                rl      =   (sbyte)buf[WorkAddress + BuffAddress + 9];
                rr      =   (sbyte)buf[WorkAddress + BuffAddress + 10];

                slope   =   (sbyte)buf[WorkAddress + BuffAddress + 11];

                trip    =   buf[WorkAddress + BuffAddress + 12];
                trip    <<= 8;
                trip    +=  buf[WorkAddress + BuffAddress + 13];

                batt    =   buf[WorkAddress + BuffAddress + 14];        //batt
                gyro    =   (sbyte)buf[WorkAddress + BuffAddress + 15]; //gyro

                side    =   (sbyte)buf[WorkAddress + BuffAddress + 16];
                side    >>= 6;
                side    &=  0x03;

                pre_sens =  (sbyte)buf[WorkAddress + BuffAddress + 16];
                pre_sens >>= 5;
                pre_sens &= 0x01;

                pos_sens =  (sbyte)buf[WorkAddress + BuffAddress + 16];
                pos_sens &= 0x1f;

                ErrorCount = (int)sens;

                log[n].mode         = mode;
                log[n].angle_t      = angle_t;
                log[n].angle        = angle;
                log[n].sv_pow       = sv_pow;
                log[n].vt           = vt;
                log[n].v            = v;
                log[n].fl           = fl;
                log[n].fr           = fr;
                log[n].rl           = rl;
                log[n].rr           = rr;
                log[n].slope_mode   = (slope >> 6) & 0x03;
                log[n].slope_sw     = (slope >> 4) & 0x03;
                log[n].slope_cnt    = slope & 0x0f;
                log[n].trip         = trip;
            //  log[n].batt         = batt;

                if(LOG_Version == 6)
                    log[n].batt     = ((float)batt)*4.0F * (float)frmOption1.nudB
att_a.Value + (float)frmOption1.nudBatt_b.Value;
                else
                    log[n].batt     = ((float)batt)*8.0F * (float)frmOption1.nudB
att_a.Value + (float)frmOption1.nudBatt_b.Value;

                log[n].gyro         = gyro;
                log[n].side         = side;
                log[n].pre_sens     = pre_sens;
                log[n].pos_sens     = pos_sens;

                //log_count         = n;

                //先読みセンサ
                if(log[n].pre_sens == 1)    log[n].sens = new StringBuilder(" P "
);
                else                        log[n].sens = new StringBuilder("   "
);

                //ラインセンサ
                if((log[n].side & 0x02) != 0) log[n].sens.Append("S");      //★
                else                          log[n].sens.Append(" ");

                for(i=0; i<8; i++){
                    switch(i){
                        case 0: case 1: case 3: case 6: case 7:
                            if((sens & 0x80) == 0)
                                log[n].sens.Append("-");
                            else
                                log[n].sens.Append("*");
                            break;
                        case 4:
                            break;
                        case 2: case 5:
                            if((sens & 0x80) == 0)
                                log[n].sens.Append("-");
                            else
                                log[n].sens.Append("+");
                            break;
                    }

                    sens <<= 1;
                }
                if((log[n].side & 0x01) != 0) log[n].sens.Append("S");      //★
                else                          log[n].sens.Append(" ");

                //ポジションセンサなし
                if(LOG_Version == 4){
                    log[n].sens.Append("      ");
                }
                //ポジションセンサあり
                else{
                    log[n].sens.Append(" ");
                    pos_sens <<= 3;
                    for(i=0; i<5; i++){
                        if((pos_sens & 0x80) == 0)
                            log[n].sens.Append("-");
                        else
                            log[n].sens.Append("*");
                        pos_sens <<= 1;
                    }
                }

                str  = new StringBuilder(String.Format("{0, 6}", time));
                time += 5;
                str.Append(String.Format("{0, 4}", log[n].mode));
                str.Append(log[n].sens);
                str.Append(" ");
                str.Append(String.Format("{0, 3}", log[n].angle_t));
                str.Append(" ");
                str.Append(String.Format("{0, 3}", log[n].angle));
                str.Append(" ");
                str.Append(String.Format("{0, 4}", log[n].sv_pow));
                str.Append(" ");
                str.Append(String.Format("{0, 4}", log[n].vt));
                str.Append(String.Format("{0, 4}", log[n].v));
                str.Append(" ");
                str.Append(String.Format("{0, 4}", log[n].fl));
                str.Append(String.Format("{0, 4}", log[n].fr));
                str.Append(String.Format("{0, 4}", log[n].rl));
                str.Append(String.Format("{0, 4}", log[n].rr));
                str.Append(String.Format("{0, 7}", log[n].trip));
                str.Append("  ");
                str.Append(String.Format("{0, 1}", log[n].slope_mode));
                str.Append(String.Format("{0, 1}", log[n].slope_sw));
                str.Append(String.Format("{0, 1}", log[n].slope_cnt));
                str.Append(String.Format("{0, 6:f1}", log[n].batt));
                str.Append(String.Format("{0, 6}", log[n].gyro));

                if (mode == -2)             //次のセクタへ
                {
                    int ii;
                    WorkAddress += 512;
                    readSize = fs.Read(buf, WorkAddress, 512);
                    BuffAddress = 0;

                    time -= 5;

                    //エラーの時はその数の分空行挿入
                    if(LOG_Version >= 2){
                        for(ii=0; ii<ErrorCount; ii++){
                            log[n].mode         = 0;
                            log[n].angle_t      = 0;
                            log[n].angle        = 0;
                            log[n].sv_pow       = 0;
                            log[n].vt           = 0;
                            log[n].v            = 0;
                            log[n].fl           = 0;
                            log[n].fr           = 0;
                            log[n].rl           = 0;
                            log[n].rr           = 0;
                            log[n].slope_mode   = 0;
                            log[n].slope_sw     = 0;
                            log[n].slope_cnt    = 0;
                            log[n].trip         = 0;
                            log[n].batt         = 0;
                            log[n].gyro         = 0;
```

```
                            log[n].side          = 0;

                            lstView.Items.Add("Err");
                            n++; if (n > 10000) break;
                        }
                    }
                }
                else
                {
                    BuffAddress += LOG_RecordBytes;
                    lstView.Items.Add(str);
                    n++; if (n > 10000) break;
                }

                if (mode == 0) break;                //modeが0なら終了
            }
        }
        //*********************************************************
        //ログバージョン004以上の読み込み終了　ここまで
        //*********************************************************

        lstView.Show();

        log_count = n;
        LogFileSize = WorkAddress + 1024;        //実質のサイズを保存用に記録して
おく

        fs.Dispose();
        menuFileSaveTXT.Enabled = true;

        DrawGraph();

        btnToubai.Enabled = true;
        btnX2.Enabled = true;
        btnX4.Enabled = true;
        btnX8.Enabled = true;


        // ハードディスクなら自動保存
        System.IO.DriveType DType;
        string drive_a, drive_b;
        drive_a = path.ToString().Substring(0,1);
        foreach(System.IO.DriveInfo DInfo in System.IO.DriveInfo.GetDrives()){
            DType = DInfo.DriveType;
            drive_b = DInfo.ToString().Substring(0,1);
            if( drive_a == drive_b ){
                if( DType == System.IO.DriveType.Fixed ){
                    //DriveIsFixedDisk = true;
                    FileSave();
                    menuFileSave.Enabled = true;
                }
                else{
                    menuFileSave.Enabled = false;
                }
            }
        }

    }


    //=====================================================================
    // Graphの描画
    //=====================================================================
    public void DrawGraph()
    {
        cur_show = false;        //カーソルを非表示に

        switch(scale_mode){
          case 1: pctGraph.Width = pnlGraph.Width;      break;
          case 2: pctGraph.Width = pnlGraph.Width * 2;  break;
          case 3: pctGraph.Width = pnlGraph.Width * 4;  break;
          case 4: pctGraph.Width = pnlGraph.Width * 8;  break;
          case 5: pctGraph.Width = log_count;           break;
          case 6: pctGraph.Width = log_count * 2;       break;
          case 7: pctGraph.Width = log_count * 4;       break;
          case 8: pctGraph.Width = log_count * 8;       break;
        }
        pctGraph.Height = pnlGraph.Height - SCROLLBAR_WIDTH;


        // PictureBoxと同サイズのBitmapオブジェクトを作成
        Bitmap bmp = new Bitmap(pctGraph.Size.Width, pctGraph.Size.Height);
        pctGraph.Image = bmp;
        Graphics g = Graphics.FromImage(pctGraph.Image);

        int n, i;
        Single x, x1;
        Pen pen_err_background = Pens.MidnightBlue;
        Pen pen_backline = Pens.DarkSlateGray;

        y0 = pctGraph.Height / 2;                                        //水
平線

        x1 = x = 0;
        graph_v = (Single)pctGraph.Width / (Single)frmMain.log_count;        //xの
増分

        for(i=y0; i<pctGraph.Height; i+=40){
            g.DrawLine(pen_backline, 0, i, pctGraph.Width, i);
        }
        for(i=y0; i>0; i-=40){
            g.DrawLine(pen_backline, 0, i, pctGraph.Width, i);
        }

        g.DrawLine(Pens.Gray,  0, y0, pctGraph.Width, y0);


        for(n=0; n<graph_points; n++){
            gp[n].y = gp[n].y1 = 0;
        }

        gp[ 0].pen = new Pen(frmOption1.lblA.ForeColor, (float)frmOption1.widthA.
Value);
        gp[ 1].pen = new Pen(frmOption1.lblB.ForeColor, (float)frmOption1.widthB.
Value);
        gp[ 2].pen = new Pen(frmOption1.lblC.ForeColor, (float)frmOption1.widthC.
Value);
        gp[ 3].pen = new Pen(frmOption1.lblD.ForeColor, (float)frmOption1.widthD.
Value);
        gp[ 4].pen = new Pen(frmOption1.lblE.ForeColor, (float)frmOption1.widthE.
Value);
        gp[ 5].pen = new Pen(frmOption1.lblF.ForeColor, (float)frmOption1.widthF.
Value);
        gp[ 6].pen = new Pen(frmOption1.lblG.ForeColor, (float)frmOption1.widthG.
Value);
        gp[ 7].pen = new Pen(frmOption1.lblH.ForeColor, (float)frmOption1.widthH.
Value);
        gp[ 8].pen = new Pen(frmOption1.lblI.ForeColor, (float)frmOption1.widthI.
Value);
        gp[ 9].pen = new Pen(frmOption1.lblJ.ForeColor, (float)frmOption1.widthJ.
Value);
        gp[10].pen = new Pen(frmOption1.lblK.ForeColor, (float)frmOption1.widthK.
Value);
        gp[11].pen = new Pen(frmOption1.lblL.ForeColor, (float)frmOption1.widthL.
Value);


        gp[ 0].enabled = frmOption1.chkA.Checked;
        gp[ 1].enabled = frmOption1.chkB.Checked;
        gp[ 2].enabled = frmOption1.chkC.Checked;
        gp[ 3].enabled = frmOption1.chkD.Checked;
        gp[ 4].enabled = frmOption1.chkE.Checked;
        gp[ 5].enabled = frmOption1.chkF.Checked;
        gp[ 6].enabled = frmOption1.chkG.Checked;
        gp[ 7].enabled = frmOption1.chkH.Checked;
        gp[ 8].enabled = frmOption1.chkI.Checked;
        gp[ 9].enabled = frmOption1.chkJ.Checked;
        gp[10].enabled = frmOption1.chkK.Checked;
        gp[11].enabled = frmOption1.chkL.Checked;

        gp[ 0].scale = (Single)frmOption1.nudA.Value;
        gp[ 1].scale = (Single)frmOption1.nudB.Value;
        gp[ 2].scale = (Single)frmOption1.nudC.Value;
        gp[ 3].scale = (Single)frmOption1.nudD.Value;
        gp[ 4].scale = (Single)frmOption1.nudE.Value;
        gp[ 5].scale = (Single)frmOption1.nudF.Value;
        gp[ 6].scale = (Single)frmOption1.nudG.Value;
        gp[ 7].scale = (Single)frmOption1.nudH.Value;
        gp[ 8].scale = (Single)frmOption1.nudI.Value;
        gp[ 9].scale = (Single)frmOption1.nudJ.Value;
        gp[10].scale = (Single)frmOption1.nudK.Value;
        gp[11].scale = (Single)frmOption1.nudL.Value;

        for(n=0; n<log_count; n++){

            if(LOG_Version <= 3){
                gp[ 0].y = -log[n].angle_t;
                gp[ 1].y = -log[n].angle;
                gp[ 2].y = -log[n].power;
                gp[ 3].y = -log[n].vt;
                gp[ 4].y = -log[n].v;
                gp[ 5].y = -log[n].batt;
                gp[ 6].y = -log[n].gyro;
                gp[ 7].y = 0;
                gp[ 8].y = 0;
                gp[ 9].y = 0;
                gp[10].y = 0;
                gp[11].y = 0;
            }
            else if(LOG_Version >= 4){
                gp[ 0].y = -log[n].angle_t;
                gp[ 1].y = -log[n].angle;
                gp[ 2].y = -log[n].sv_pow;
                gp[ 3].y = -log[n].vt;
                gp[ 4].y = -log[n].v;
                gp[ 5].y = -log[n].fl;
                gp[ 6].y = -log[n].fr;
                gp[ 7].y = -log[n].rl;
                gp[ 8].y = -log[n].rr;
                gp[ 9].y = -log[n].trip;
                gp[10].y = -log[n].batt;
                gp[11].y = -log[n].gyro;
            }
            for(i=0; i<12; i++){
                gp[i].y = gp[i].y * gp[i].scale * (Single)y0 / 1000;
            }

            x += graph_v;


            if(log[n].mode == 0){        //ログ記録エラーの部分は背景を替えてグラ
フ描画はしない。
                int ix;

                for(ix = (int)x1; ix < x; ix++){
                    g.DrawLine(pen_err_background, ix, 0, ix, (int)pctGraph.Heigh
t);
                }
            }
            else{                        //通常描画
                if(n % 20 == 0){         //縦線
                    g.DrawLine(pen_backline, x1, 0, x1, pctGraph.Height);
                }

                for(i=11; i>=0; i--){
                    if(gp[i].enabled == true){
                        if(n>0) if(log[n-1].mode == 0) gp[i].y1 = gp[i].y;

                        g.DrawLine(gp[i].pen, x, gp[i].y + y0, x1, gp[i].y1 + y0)
;
                        gp[i].y1 = gp[i].y;
                    }
                }
            }
            x1 = x;
        }

        pctGraph.Refresh();     // PictureBoxを更新（再描画させる）


        for(i=0; i<graph_points; i++){
            gp[i].pen.Dispose();
        }
        g.Dispose();
    }

    //=====================================================================
    //現在位置のカーソルを消去
    //=====================================================================
    private void erase_cursol()
    {
        Point p1, p2, ps, pe;
        Point pgx = pnlGraph.PointToScreen(new Point(0, 0));

        //現在のカーソルを消去
        p1 = new Point((int)cur_x1, 0);
        p2 = new Point((int)cur_x1, pctGraph.Height);
        ps = pctGraph.PointToScreen(p1);
        pe = pctGraph.PointToScreen(p2);
        if(ps.X > pgx.X && ps.X < pgx.X + pnlGraph.Width){
            ControlPaint.DrawReversibleLine(ps, pe, Color.Black);
```

```
        }
    }

    //=========================================================================
    //新しい位置にカーソルを表示
    //=========================================================================
    private void draw_cursol()
    {
        Point p1, p2, ps, pe;
        Point pgx = pnlGraph.PointToScreen(new Point(0, 0));
        int n;

        if(cur_show){       //カーソルが表示されていたら現在のカーソルを消去
            erase_cursol();
        }
        else{
            cur_show = true;
        }

        //新しい場所の位置を計算
        n = lstView.SelectedIndex;
        cur_x = (int)((Single)n * graph_v);

        //新しい場所にカーソル表示
        p1 = new Point((int)cur_x, 0);
        p2 = new Point((int)cur_x, pctGraph.Height);
        ps = pctGraph.PointToScreen(p1);
        pe = pctGraph.PointToScreen(p2);
        if(ps.X > pgx.X && ps.X < pgx.X + pnlGraph.Width){
            ControlPaint.DrawReversibleLine(ps, pe, Color.Black);
        }

        //新しい場所をcur_x1に記録
        cur_n1 = n;
        cur_x1 = cur_x;
    }


    //=========================================================================
    // 縦カーソルの描画
    //=========================================================================
    private void lstView_SelectedIndexChanged(object sender, EventArgs e)
    {
        draw_cursol();
    }

    //=========================================================================
    //画面上のグラフカーソル描画
    //=========================================================================
    private void pctGraph_Paint(object sender, PaintEventArgs e)
    {
        if(cur_show){
            erase_cursol();
            cur_show = false;
        }
    }

    //=========================================================================
    //ファイルを開く（メニュー及びコマンドボタンより）
    //=========================================================================
    private void FileOpen_Click(object sender, EventArgs e)
    {
        //“開く”ダイアログボックス
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.InitialDirectory = txtPath.Text;
        ofd.Filter = "MCRログファイル (*.LOG)|*.LOG|" + "すべてのファイル (*.*)|*
.*";
        ofd.FilterIndex = 1;
        ofd.Multiselect = false;
        if (ofd.ShowDialog() == DialogResult.OK){
            FileOpen(ofd.FileName);
        }
        ofd.Dispose();
    }

    //=========================================================================
    //ファイルの保存
    //=========================================================================
    private void menuFileSave_Click(object sender, EventArgs e)
    {
        FileSave();
    }

    //=========================================================================
    //フォームのリサイズ
    //=========================================================================
    private void frmMain_Resize(object sender, EventArgs e)
    {
        try{
            if(lblHead1.Size.Width > 0){
                splitContainer1.SplitterDistance = lblHead1.Size.Width + SCROLLBA
R_WIDTH;
            }
        }
        catch(Exception){
        }
    }

    //=========================================================================
    //スプリットバー操作
    //=========================================================================
    private void splitContainer2_Panel1_Resize(object sender, EventArgs e)
    {
        txtHead.Height = splitContainer2.Panel1.Height - txtHead.Location.Y;
    }

    private void splitContainer2_SizeChanged(object sender, EventArgs e)
    {
        txtHead.Width = splitContainer2.Panel1.Width - txtHead.Location.X * 3;
        txtPath.Width = txtHead.Width;
        lstView.Width = txtHead.Width;
        lstView.Height = splitContainer2.Panel2.Height - lstView.Location.Y;
    }

    private void splitContainer1_Panel2_Resize(object sender, EventArgs e)
    {
        pnlGraph.Width = splitContainer1.Panel2.Width - SCROLLBAR_WIDTH;
        pnlGraph.Height = splitContainer1.Panel2.Height - pnlGraph.Top - SCROLLBA
R_WIDTH;
    }

    //=========================================================================
    //エクスプローラからのファイルドラッグエンター
    //=========================================================================
    private void frmMain_DragEnter(object sender, DragEventArgs e)
    {
        e.Effect = DragDropEffects.All;
    }

    //=========================================================================
    //エクスプローラからのファイルドロップ
    //=========================================================================
    private void frmMain_DragDrop(object sender, DragEventArgs e)
    {
        if (e.Data.GetDataPresent(DataFormats.FileDrop)){
            foreach (string fileName in (string[])e.Data.GetData(DataFormats.File
Drop)){
                FileOpen(fileName);
            }
        }
    }

    //=========================================================================
    //メインフォームのコンストラクタ
    //=========================================================================
    public frmMain()
    {
        InitializeComponent();
    }

    //=========================================================================
    //メインフォームの起動
    //=========================================================================
    private void frmMain_Load(object sender, EventArgs e)
    {
    //  //多重起動の禁止
    //  mut = new System.Threading.Mutex(false, "myMutex");
    //  if(mut.WaitOne(0, false) == false){
    //      this.Close();
    //  }

        //起動時のファイル名取得
        string[] cmds = System.Environment.GetCommandLineArgs();
        if(cmds.Length > 1){
            for( int i=1; i < cmds.Length; i++ ){
                FileOpen(cmds[i]);
            }
        }

        //PictureBoxのサイズ設定
        pctGraph.Width = pnlGraph.Width;
        pctGraph.Height = pnlGraph.Height;
    }

    //=========================================================================
    //メインフォームの終了
    //=========================================================================
    private void frmMain_FormClosed(object sender, FormClosedEventArgs e)
    {
        //mut.Close();
    }

    //=========================================================================
    //テキスト形式でファイルセーブ
    //=========================================================================
    private void menuFileSaveTXT_Click(object sender, EventArgs e)
    {
        string path_txt;
        int i, n=0;

        if (path == ""){
            MessageBox.Show("ファイルがありません");
            return;
        }

        if(lstView.Items.Count == 0){
            return;
        }

        path_txt = path.Substring(0, path.Length - 3) + "TXT";
        //“保存”ダイアログボックス
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.InitialDirectory = path_txt;// txtPath.Text;
        sfd.FileName = path_txt;
        sfd.Filter = "MCR TXTログファイル (*.TXT)|*.TXT|" + "すべてのファイル (*.
*)|*.*";
        sfd.FilterIndex = 1;
        if (sfd.ShowDialog() == DialogResult.OK){
            path_txt = sfd.FileName;
        }
        else{
            sfd.Dispose();
            return;
        }
        sfd.Dispose();

        //
        // テキストデータの書き込み
        //
        System.IO.StreamWriter TextFile;
        TextFile = new System.IO.StreamWriter(path_txt);
        if(txtHead.Text.Length > 0){
            TextFile.WriteLine(txtHead.Text);
        }

        //TextFile.WriteLine("mode  sens   hnd ang  pow  vt   v slc    trip  diff
    z      x");
        //TextFile.WriteLine("----------------------------------------------------
-------------");

        TextFile.WriteLine(lblHead1.Text);
        for(i = 0; i < lblHead1.Text.Length; i++){
            TextFile.Write("-");
        }
        TextFile.WriteLine();

    for(n=0; n<lstView.Items.Count; n++){
            TextFile.WriteLine(lstView.Items[n]);
        }
        TextFile.Close();
        TextFile.Dispose();
        MessageBox.Show(path_txt, "書き込み終了");
    }

    //=========================================================================
    //メニュー：終了
    //=========================================================================
    private void menuFileExit_Click(object sender, EventArgs e)
    {
        this.Close();
```

```
        }

        //=====================================================================
        //グラフ更新
        //=====================================================================
        private void btnOK_Click(object sender, EventArgs e)
        {
            DrawGraph();
        }

        //=====================================================================
        //グラフ縮尺関連
        //=====================================================================
        private void btnToubai_Click(object sender, EventArgs e){
            scale_mode = 5;
            DrawGraph();
        }
        private void btnX2_Click(object sender, EventArgs e){
            scale_mode = 6;
            DrawGraph();
        }
        private void btnX4_Click(object sender, EventArgs e){
            scale_mode = 7;
            DrawGraph();
        }

        private void btnX8_Click(object sender, EventArgs e){
            scale_mode = 8;
            DrawGraph();
        }

        private void btnX1_Click(object sender, EventArgs e){
            scale_mode = 1;
            DrawGraph();
        }

        private void btnW2_Click(object sender, EventArgs e){
            scale_mode = 2;
            DrawGraph();
        }

        private void btnW4_Click(object sender, EventArgs e){
            scale_mode = 3;
            DrawGraph();
        }

        private void btnW8_Click(object sender, EventArgs e){
            scale_mode = 4;
            DrawGraph();
        }

        //=====================================================================
        //オプションボタン
        //=====================================================================
        private void btnGraphOption_Click(object sender, EventArgs e)
        {
            if (frmOption1.ShowDialog() == DialogResult.OK){
                DrawGraph();
            }
        }

        //=====================================================================
        //グラフのクリックでlstViewのインデックス変更
        //=====================================================================
        private void pctGraph_MouseMove(object sender, MouseEventArgs e)
        {
            if(e.Button == MouseButtons.Left){
                if(lstView.Items.Count > 0){
                    int x = (int)(e.X / graph_v);
                    if(x < 0)
                        x = 0;
                    else if(x >= lstView.Items.Count)
                        x = lstView.Items.Count - 1;
                    lstView.SelectedIndex = x;
                    lstView.Focus();
                }
            }
            else if(e.Button == MouseButtons.Right){
                Point pnt2 = new Point(e.X, e.Y);
                pnt2 = pctGraph.PointToScreen(pnt2);
                int x = pnt2.X - scrPoint2.X;
                int y = pnt2.Y - scrPoint2.Y;
                pnlGraph.AutoScrollPosition = new Point(-scrPoint1.X + x * -1, -scrPo
int1.Y + y * -1);
            }
        }

        private void pctGraph_MouseDown(object sender, MouseEventArgs e)
        {
            if(e.Button == MouseButtons.Left){
                if(lstView.Items.Count > 0){
                    int x = (int)(e.X / graph_v);
                    if(x < 0)
                        x = 0;
                    else if(x >= lstView.Items.Count)
                        x = lstView.Items.Count - 1;
                    lstView.SelectedIndex = x;
                    lstView.Focus();
                }
            }
            else if(e.Button == MouseButtons.Right){
                scrPoint1 = pnlGraph.AutoScrollPosition;
                scrPoint2 = new Point(e.X, e.Y);
                scrPoint2 = pctGraph.PointToScreen(scrPoint2);

                if(cur_show){
                    erase_cursol();
                    cur_show = false;
                }
            }
        }

        private void pnlGraph_Scroll(object sender, ScrollEventArgs e)
        {
            if(cur_show){
                erase_cursol();
                cur_show = false;
            }
        }
    }

    //=====================================================================
    //二重起動の禁止と、最初のインスタンスに後で起動した引数を渡す処理
    //=====================================================================
    class myApplication:WindowsFormsApplicationBase{
        public myApplication() : base() {
            this.EnableVisualStyles = true;
            this.IsSingleInstance = true;
            this.MainForm = new frmMain();//スタートアップフォームを設定
            this.StartupNextInstance += new StartupNextInstanceEventHandler(myApplica
tion_StartupNextInstance);
        }
        void myApplication_StartupNextInstance(object sender, StartupNextInstanceEven
tArgs e) {
            //ここに二重起動されたときの処理を書く
            //e.CommandLineでコマンドライン引数を取得出来る
            frmMain frmMain1 = (frmMain)MainForm;

            //起動時のファイル名取得
            foreach(string cmd in e.CommandLine){
                frmMain1.FileOpen(cmd);
            }
        }
    }
}
```