**Общество с ограниченной ответственностью «ДЕЦИМА»**

УТВЕРЖДАЮ

Директор ООО «ДЕЦИМА»

/Шкляев А.А./

13.04. 2020г.

**ПРОГРАММА ДЛЯ ЭВМ**

**Программа-конфигуратор**

**УРОВ на ПК**

**Авторы**: Порохня Дмитрий Викторович

Исходный текст программы

75 Листов

© ООО «ДЕЦИМА» 2020

**Москва 2020г.**

**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**.**Ports**;**

**using** System**.**Globalization**;**

**namespace** UROVConfig

**{**

**public** partial class MainForm **:** Form

**{**

**private** ToolStripMenuItem lastSelectedPort **=** **null;**

**private** ITransport currentTransport **=** **null;**

**private** ConnectForm connForm **=** **null;**

**private** const char PARAM\_DELIMITER **=** '|'**;**

**private** const string GET\_PREFIX **=** "GET="**;**

**private** const string SET\_PREFIX **=** "SET="**;**

**public** MainForm**()**

**{**

InitializeComponent**();**

**}**

**private** void exitToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**

**{**

Application**.**Exit**();**

**}**

/// <summary>

/// закрываем форму ожидания соединения

/// </summary>

**private** void EnsureCloseConnectionForm**()**

**{**

**if** **(**connForm **!=** **null)**

**{**

connForm**.**DialogResult **=** System**.**Windows**.**Forms**.**DialogResult**.**OK**;**

**}**

connForm **=** **null;**

**}**

void OnTryToConnectToPort**(**string portname**)**

**{**

**if** **(**InvokeRequired**)**

**{**

Invoke**((**MethodInvoker**)delegate** **{** OnTryToConnectToPort**(**portname**);** **});**

**return;**

**}**

**if** **(**connForm **!=** **null)**

**{**

connForm**.**lblCurrentAction**.**Text **=** "Соединяемся с портом " **+** portname **+** "..."**;**

**}**

**}**

void OnTransportDisconnect**(**ITransport transport**)**

**{**

**if** **(**InvokeRequired**)**

**{**

Invoke**((**MethodInvoker**)delegate** **{** OnTransportDisconnect**(**transport**);** **});**

**return;**

**}**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"TRANSPORT DISCONNECT EVENT!"**);**

**this.**InitAfterConnect**(false);**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"TRANSPORT DISCONNECT EVENT DONE!"**);**

**}**

/// <summary>

/// Коннектимся к порту

/// </summary>

/// <param name="port">Имя порта</param>

**public** void StartConnectToPort**(**string port**,** bool withHandshake**,** bool findDevice**)**

**{**

//System.Diagnostics.Debug.Assert(currentTransport == null);

System**.**Diagnostics**.**Debug**.**WriteLine**(**"START CONNECT, HANDLER IN MAIN FORM..."**);**

**if** **(**currentTransport **!=** **null)**

**{**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"CLOSE EXISTING CONNECTION..."**);**

currentTransport**.**Disconnect**();**

//TODO: тут надо дождаться, когда транспорт полностью освободится !!!!

**while(**currentTransport**.**Connected**())**

**{**

Application**.**DoEvents**();**

**}**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"EXISTING CONNECTION CLOSED!"**);**

**}**

int speed **=** GetConnectionSpeed**();**

// создаём новый транспорт

currentTransport **=** **new** SerialPortTransport**(**port**,**speed**);**

currentTransport**.**OnConnect **=** **new** ConnectResult**(**OnCOMConnect**);**

currentTransport**.**OnDataReceived **=** **new** TransportDataReceived**(**OnDataFromCOMPort**);**

currentTransport**.**OnDisconnect **=** **new** TransportDisconnect**(**OnTransportDisconnect**);**

currentTransport**.**OnTryToConnectToPort **=** **new** TryToConnectToPort**(**OnTryToConnectToPort**);**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"START TRANSPORT CONNECT...."**);**

// коннектимся

currentTransport**.**Connect**(**withHandshake**,** findDevice**);**

**}**

/// <summary>

/// Обрабатываем строку, пришедшую из транспорта COM-порта

/// </summary>

/// <param name="line"></param>

**private** void OnDataFromCOMPort**(**byte**[]** data**)**

**{**

**if** **(**InvokeRequired**)**

**{**

Invoke**((**MethodInvoker**)delegate** **{** OnDataFromCOMPort**(**data**);** **});**

**return;**

**}**

EnsureCloseConnectionForm**();** // закрываем форму коннекта, если она ещё не закрыта

// обрабатываем данные, полученные из порта

//ProcessPortAnswer(line.Trim());

ProcessPortAnswer**(**data**);**

**}**

// Буфер под ответ с SD-карты

**private** List**<**byte**>** SDQueryAnswer **=** **new** List**<**byte**>();**

/// <summary>

/// Буфер под ответ с COM-порта

/// </summary>

//string COMBuffer = "";

**private** void ProcessAnswerLine**(**string line**)**

**{**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"<= COM: " **+** line**);**

**if** **(**line**.**StartsWith**(**"UROV v."**))** // нашли загрузку ядра, следовательно, можно писать данные

**{**

tbFirmwareVersion**.**Text **=** line**;**

coreBootFound **=** **true;**

**}**

bool isKnownAnswer **=** line**.**StartsWith**(**"OK="**)** **||** line**.**StartsWith**(**"ER="**);**

**this.**AddToLog**(**line**,** isKnownAnswer**);** // добавляем данные в лог

**if** **(!**isKnownAnswer**)** // нам тут ловить нечего

**{**

**return;**

**}**

**if** **(this.**currentCommand**.**ParseFunction **!=** **null)**

**{**

Answer a **=** **new** Answer**(**line**);**

**this.**currentCommand**.**ParseFunction**(**a**);**

**this.**currentCommand**.**ParseFunction **=** **null;** // освобождаем

**this.**currentCommand**.**CommandToSend **=** ""**;**

**}**

**}**

// private FileDownloadFlags fileDownloadFlags = FileDownloadFlags.View;

**private** List**<**byte**>** logContentToShow **=** **null;**

**private** void DoShowLogFile**(**ConnectForm frm**)**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**logContentToShow **!=** **null);**

ShowLogFile**(**logContentToShow**,** **this.**logDataGrid**,** ""**,** **true,**frm**,null,false);**

frm**.**DialogResult **=** DialogResult**.**OK**;**

**}**

**private** void ViewFile**(**List**<**byte**>** content**)**

**{**

ShowWaitCursor**(false);**

statusProgressBar**.**Visible **=** **false;**

statusProgressMessage**.**Visible **=** **false;**

**this.**btnViewSDFile**.**Enabled **=** treeViewSD**.**SelectedNode **!=** **null;**

**this.**btnDeleteSDFile**.**Enabled **=** treeViewSD**.**SelectedNode **!=** **null;**

**this.**btnListSDFiles**.**Enabled **=** **true;**

**this.**treeViewSD**.**Enabled **=** **true;**

string upStr **=** **this.**requestedFileName**.**ToUpper**();**

**if** **(**upStr**.**EndsWith**(**".LOG"**))**

**{**

ConnectForm cn **=** **new** ConnectForm**(true);**

cn**.**OnConnectFormShown **=** **this.**DoShowLogFile**;**

cn**.**lblCurrentAction**.**Text **=** "Загружаем лог..."**;**

logContentToShow **=** content**;**

cn**.**ShowDialog**();**

**}**

**else** **if** **(**upStr**.**EndsWith**(**".ETL"**))**

**{**

CreateChart**(**content**,** **this.**ethalonChart**);**

**this.**plEthalonChart**.**BringToFront**();**

**}**

**}**

**private** void SaveEthalonFile**(**List**<**byte**>** content**)**

**{**

SaveEthalon**(**requestedFileName**,** content**);**

**}**

/\*

/// <summary>

/// Показываем содержимое файла

/// </summary>

/// <param name="content"></param>

private void DealWithFileContent(List<byte> content)

{

switch (fileDownloadFlags)

{

case FileDownloadFlags.View:

{

ShowWaitCursor(false);

statusProgressBar.Visible = false;

statusProgressMessage.Visible = false;

this.btnViewSDFile.Enabled = treeViewSD.SelectedNode != null;

this.btnDeleteSDFile.Enabled = treeViewSD.SelectedNode != null;

this.btnListSDFiles.Enabled = true;

this.treeViewSD.Enabled = true;

string upStr = this.requestedFileName.ToUpper();

if (upStr.EndsWith(".LOG"))

{

ShowLogFile(content);

}

else if (upStr.EndsWith(".ETL"))

{

CreateEthalonChart(content);

this.plEthalonChart.BringToFront();

}

}

break;

case FileDownloadFlags.DownloadEthalon:

{

SaveEthalon(requestedFileName, content);

fileDownloadFlags = FileDownloadFlags.View;

}

break;

} // switch

}

\*/

**private** List**<**int**>** ethalon0UpData **=** **new** List**<**int**>();**

**private** List**<**int**>** ethalon0DwnData **=** **new** List**<**int**>();**

**private** List**<**int**>** ethalon1UpData **=** **new** List**<**int**>();**

**private** List**<**int**>** ethalon1DwnData **=** **new** List**<**int**>();**

**private** List**<**int**>** ethalon2UpData **=** **new** List**<**int**>();**

**private** List**<**int**>** ethalon2DwnData **=** **new** List**<**int**>();**

**private** void ClearEthalons**()**

**{**

ethalon0UpData**.**Clear**();**

ethalon0DwnData**.**Clear**();**

ethalon1UpData**.**Clear**();**

ethalon1DwnData**.**Clear**();**

ethalon2UpData**.**Clear**();**

ethalon2DwnData**.**Clear**();**

**this.**requestEthalonCounter **=** 0**;**

**}**

**private** UInt16 Read16**(**List**<**byte**>** content**,** int idx**)**

**{**

UInt16 result **=** content**[**idx **+** 1**];**

result **<<=** 8**;**

result **|=** content**[**idx**];**

**return** result**;**

**}**

**private** int Read32**(**List**<**byte**>** content**,** int idx**)**

**{**

int result **=** **(**content**[**idx **+** 3**]** **<<** 24**)** **|** **(**content**[**idx **+** 2**]** **<<** 16**)** **|** **(**content**[**idx **+** 1**]** **<<** 8**)** **|** content**[**idx**];**

**return** result**;**

**}**

**public** class LogInfo

**{**

**public** List**<**InterruptRecord**>** list **=** **new** List**<**InterruptRecord**>();**

**public** string addToColumnName**;**

**public** bool computeMotoresurcePercents**;**

**}**

**private** Dictionary**<**DataGridView**,** LogInfo**>** gridToListCollection **=** **new** Dictionary**<**DataGridView**,** LogInfo**>();**

**private** void ShowLogFile**(**List**<**byte**>** content**,** DataGridView targetGrid**,** string addToColumnName**,** bool computeMotoresurcePercents**,** ConnectForm frm**,** ShowInterruptInfo callback**,** bool stopAfterFirstRecord**)**

**{**

**if** **(**targetGrid **!=** **null)**

**{**

ClearInterruptsList**(**targetGrid**);**

**}**

// парсим лог-файл

int readed **=** 0**;**

InterruptInfo currentInterruptInfo **=** **null;**

InterruptRecord curRecord **=** **null;**

**try**

**{**

bool stopped **=** **false;**

**while** **(**readed **<** content**.**Count **&&** **!**stopped**)**

**{**

**if** **(**frm **!=** **null)**

**{**

string message **=** "Загружаем лог: {0}% из {1} байт..."**;**

int percents **=** **(**readed **\*** 100**)** **/** content**.**Count**;**

frm**.**lblCurrentAction**.**Text **=** String**.**Format**(**message**,** percents**,** content**.**Count**);**

**}**

Application**.**DoEvents**();**

byte curByte **=** content**[**readed**];**

readed**++;**

LogRecordType recType **=** **(**LogRecordType**)**curByte**;**

**switch** **(**recType**)**

**{**

**case** LogRecordType**.**InterruptInfoBegin**:**

**{**

currentInterruptInfo **=** **new** InterruptInfo**();**

**}**

**break;**

**case** LogRecordType**.**InterruptTime**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**currentInterruptInfo **!=** **null);**

// далее идут 7 байт времени

byte dayOfMonth **=** content**[**readed**];** readed**++;**

byte month **=** content**[**readed**];** readed**++;**

UInt16 year **=** Read16**(**content**,** readed**);** readed **+=** 2**;**

byte hour **=** content**[**readed**];** readed**++;**

byte minute **=** content**[**readed**];** readed**++;**

byte second **=** content**[**readed**];** readed**++;**

currentInterruptInfo**.**InterruptTime **=** **new** DateTime**(**year**,** month**,** dayOfMonth**,** hour**,** minute**,** second**);**

**}**

**break;**

**case** LogRecordType**.**SystemTemperature**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**currentInterruptInfo **!=** **null);**

// далее идут два байта температуры

byte value **=** content**[**readed**];** readed**++;**

byte fract **=** content**[**readed**];** readed**++;**

float fVal **=** value **\*** 100 **+** fract**;**

fVal **/=** 100**;**

currentInterruptInfo**.**SystemTemperature **=** fVal**;**

**}**

**break;**

**case** LogRecordType**.**InterruptRecordBegin**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**currentInterruptInfo **!=** **null);**

curRecord **=** **new** InterruptRecord**();**

curRecord**.**InterruptInfo **=** currentInterruptInfo**;**

**}**

**break;**

**case** LogRecordType**.**ChannelNumber**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// далее идёт байт номера канала

curRecord**.**ChannelNumber **=** content**[**readed**];** readed**++;**

**}**

**break;**

**case** LogRecordType**.**ChannelInductiveSensorState**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// далее идёт байт номера канала

//DEPRECATED: curRecord.InductiveSensorState = (InductiveSensorState) content[readed]; readed++;

readed**++;**

**}**

**break;**

**case** LogRecordType**.**RodPosition**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// далее идёт позиция штанги

curRecord**.**RodPosition **=** **(**RodPosition**)**content**[**readed**];** readed**++;**

**}**

**break;**

**case** LogRecordType**.**MoveTime**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// далее идут четыре байта времени движения штанги

curRecord**.**MoveTime **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

**}**

**break;**

**case** LogRecordType**.**RelayTriggeredTime**:** // время срабатывания защиты

**{**

// далее идут 7 байт времени

byte dayOfMonth **=** content**[**readed**];** readed**++;**

byte month **=** content**[**readed**];** readed**++;**

UInt16 year **=** Read16**(**content**,** readed**);** readed **+=** 2**;**

byte hour **=** content**[**readed**];** readed**++;**

byte minute **=** content**[**readed**];** readed**++;**

byte second **=** content**[**readed**];** readed**++;**

**}**

**break;**

**case** LogRecordType**.**Motoresource**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// далее идут 4 байта моторесурса

curRecord**.**Motoresource **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

**}**

**break;**

**case** LogRecordType**.**EthalonNumber**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// байт номера эталона

curRecord**.**EthalonCompareNumber **=** **(**EthalonCompareNumber**)**content**[**readed**];** readed**++;**

**}**

**break;**

**case** LogRecordType**.**CompareResult**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// байт результатов сравнения с эталоном

curRecord**.**EthalonCompareResult **=** **(**EthalonCompareResult**)**content**[**readed**];** readed**++;**

**}**

**break;**

**case** LogRecordType**.**EthalonDataFollow**:**

**{**

// следом идут данные эталона, с которым сравнивали

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

curRecord**.**EthalonData**.**Clear**();**

// следом идут 2 байта длины данных

int dataLen **=** Read16**(**content**,** readed**);** readed **+=** 2**;**

// далее идут пачки по 4 байта записей по прерыванию

**for** **(**int k **=** 0**;** k **<** dataLen**;** k**++)**

**{**

int curInterruptData **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

curRecord**.**EthalonData**.**Add**(**curInterruptData**);**

**}** // for

/\*

// добавляем фейковую нулевую точку, дублированием первой.

// между ними, таким образом, окажется нулевой промежуток времени.

if (curRecord.EthalonData.Count > 0)

{

int first = curRecord.EthalonData[0];

curRecord.EthalonData.Insert(0, first);

}

\*/

**}**

**break;**

**case** LogRecordType**.**DataArrivedTime**:** // смещение от начала данных по току до начала данных по прерываниям

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// далее идёт смещение, в миллисекундах, 4 байта

curRecord**.**DataArrivedTime **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

**}**

**break;**

**case** LogRecordType**.**OscDataFollow**:** // идут данные по току для канала

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

curRecord**.**CurrentTimes**.**Clear**();**

curRecord**.**CurrentData1**.**Clear**();**

curRecord**.**CurrentData2**.**Clear**();**

curRecord**.**CurrentData3**.**Clear**();**

// следом идут 2 байта длины данных

int dataLen **=** Read16**(**content**,** readed**);** readed **+=** 2**;**

// далее идут пачки по 4 байта записей по времени сбора записей по току

**for** **(**int k **=** 0**;** k **<** dataLen**;** k**++)**

**{**

int curData **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

curRecord**.**CurrentTimes**.**Add**(**curData**);**

**}** // for

// далее идут пачки по 4 байта записей по току канала 1

**for** **(**int k **=** 0**;** k **<** dataLen**;** k**++)**

**{**

int curData **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

curRecord**.**CurrentData1**.**Add**(**curData**);**

**}** // for

// далее идут пачки по 4 байта записей по току канала 2

**for** **(**int k **=** 0**;** k **<** dataLen**;** k**++)**

**{**

int curData **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

curRecord**.**CurrentData2**.**Add**(**curData**);**

**}** // for

// далее идут пачки по 4 байта записей по току канала 3

**for** **(**int k **=** 0**;** k **<** dataLen**;** k**++)**

**{**

int curData **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

curRecord**.**CurrentData3**.**Add**(**curData**);**

**}** // for

**}**

**break;**

**case** LogRecordType**.**InterruptDataBegin**:**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

// начало данных по прерыванию

curRecord**.**InterruptData**.**Clear**();**

// следом идут 2 байта длины данных

int dataLen **=** Read16**(**content**,** readed**);** readed **+=** 2**;**

// далее идут пачки по 4 байта записей по прерыванию

**for** **(**int k **=** 0**;** k **<** dataLen**;** k**++)**

**{**

int curInterruptData **=** Read32**(**content**,** readed**);** readed **+=** 4**;**

curRecord**.**InterruptData**.**Add**(**curInterruptData**);**

**}** // for

/\*

// добавляем фейковую нулевую точку, дублированием первой.

// между ними, таким образом, окажется нулевой промежуток времени.

if(curRecord.InterruptData.Count > 0)

{

int first = curRecord.InterruptData[0];

curRecord.InterruptData.Insert(0, first);

}

\*/

**}**

**break;**

**case** LogRecordType**.**InterruptDataEnd**:**

**{**

// конец данных

**}**

**break;**

**case** LogRecordType**.**InterruptRecordEnd**:**

**{**

**if** **(**targetGrid **!=** **null)**

**{**

AddInterruptRecordToList**(**curRecord**,** targetGrid**,** addToColumnName**,** computeMotoresurcePercents**);**

**}**

**if** **(**stopAfterFirstRecord**)**

**{**

stopped **=** **true;**

**}**

**if** **(**callback **!=** **null)**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**curRecord **!=** **null);**

callback**(**curRecord**);**

**}**

**}**

**break;**

**case** LogRecordType**.**InterruptInfoEnd**:**

**{**

currentInterruptInfo **=** **null;**

**}**

**break;**

**}** // switch

**}** // while

**}**

**catch**

**{**

**if** **(**targetGrid **!=** **null)**

**{**

**this.**plEmptySDWorkspace**.**BringToFront**();**

**}**

MessageBox**.**Show**(**"Ошибка разбора лог-файла!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**return;**

**}**

**if** **(**targetGrid **!=** **null)**

**{**

**if** **(**gridToListCollection**.**ContainsKey**(**targetGrid**))**

**{**

targetGrid**.**RowCount **=** gridToListCollection**[**targetGrid**].**list**.**Count**;**

**}**

targetGrid**.**BringToFront**();**

**}**

**}**

**private** void AddInterruptRecordToList**(**InterruptRecord record**,** DataGridView targetGrid**,** string addToColumnName**,** bool computeMotoresurcePercents**)**

**{**

**if** **(**record **==** **null)**

**return;**

**if(!**gridToListCollection**.**ContainsKey**(**targetGrid**))**

**{**

LogInfo linf **=** **new** LogInfo**();**

linf**.**addToColumnName **=** addToColumnName**;**

linf**.**computeMotoresurcePercents **=** computeMotoresurcePercents**;**

gridToListCollection**[**targetGrid**]** **=** linf**;**

**}**

//Тут добавление в список в таблицу

gridToListCollection**[**targetGrid**].**list**.**Add**(**record**);**

**}**

**private** void ClearInterruptsList**(**DataGridView targetGrid**)**

**{**

// Тут очистка таблицы

targetGrid**.**RowCount **=** 0**;**

**if** **(**gridToListCollection**.**ContainsKey**(**targetGrid**))**

**{**

gridToListCollection**[**targetGrid**].**list**.**Clear**();**

**}**

**}**

**private** void SaveEthalon**(**string fname**,** List**<**byte**>** content**)**

**{**

byte**[]** dt **=** **new** byte**[**4**];**

List**<**int**>** timeList **=** **new** List**<**int**>();**

**for** **(**int i **=** 0**;** i **<** content**.**Count**;** i **+=** 4**)**

**{**

**try**

**{**

dt**[**0**]** **=** content**[**i**];**

dt**[**1**]** **=** content**[**i **+** 1**];**

dt**[**2**]** **=** content**[**i **+** 2**];**

dt**[**3**]** **=** content**[**i **+** 3**];**

int curVal **=** BitConverter**.**ToInt32**(**dt**,** 0**);**

timeList**.**Add**(**curVal**);**

**}**

**catch**

**{**

**break;**

**}**

**}**

**if** **(**fname**.**EndsWith**(**"ET0UP.ETL"**))**

ethalon0UpData **=** timeList**;**

**else**

**if** **(**fname**.**EndsWith**(**"ET0DWN.ETL"**))**

ethalon0DwnData **=** timeList**;**

**else**

**if** **(**fname**.**EndsWith**(**"ET1UP.ETL"**))**

ethalon1UpData **=** timeList**;**

**else**

**if** **(**fname**.**EndsWith**(**"ET1DWN.ETL"**))**

ethalon1DwnData **=** timeList**;**

**else**

**if** **(**fname**.**EndsWith**(**"ET2UP.ETL"**))**

ethalon2UpData **=** timeList**;**

**else**

**if** **(**fname**.**EndsWith**(**"ET2DWN.ETL"**))**

**{**

ethalon2DwnData **=** timeList**;**

ethalonsRequested **=** **true;**

**}**

**}**

**private** void CreateChart**(**List**<**byte**>** content**,** System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**Chart targetChart**)**

**{**

System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**Series s **=** targetChart**.**Series**[**0**];**

s**.**Points**.**Clear**();**

// у нас размер одной записи - 4 байта

int pointsCount **=** content**.**Count **/** 4**;**

byte**[]** dt **=** **new** byte**[**4**];**

int xStep **=** 1**;**

// подсчитываем максимальное значение по Y

List**<**int**>** timeList **=** **new** List**<**int**>();**

**for** **(**int i **=** 0**;** i **<** content**.**Count**;** i **+=** 4**)**

**{**

**try**

**{**

dt**[**0**]** **=** content**[**i**];**

dt**[**1**]** **=** content**[**i **+** 1**];**

dt**[**2**]** **=** content**[**i **+** 2**];**

dt**[**3**]** **=** content**[**i **+** 3**];**

int curVal **=** BitConverter**.**ToInt32**(**dt**,** 0**);**

timeList**.**Add**(**curVal**);**

**}**

**catch**

**{**

**break;**

**}**

**}**

// получаем максимальное время импульса - это будет 100% по оси Y

int maxPulseTime **=** 0**;**

**for** **(**int i **=** 1**;** i **<** timeList**.**Count**;** i**++)**

**{**

maxPulseTime **=** Math**.**Max**(**maxPulseTime**,** **(**timeList**[**i**]** **-** timeList**[**i **-** 1**]));**

**}**

double xCoord **=** 0**;**

// теперь считаем все остальные точки

**for** **(**int i **=** 1**;** i **<** timeList**.**Count**;** i**++)**

**{**

int pulseTime **=** timeList**[**i**]** **-** timeList**[**i **-** 1**];**

pulseTime **\*=** 100**;**

int pulseTimePercents **=** pulseTime **/** maxPulseTime**;**

pulseTimePercents **=** 100 **-** pulseTimePercents**;**

System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint pt **=** **new** System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint**();**

pt**.**XValue **=** xCoord**;**

pt**.**SetValueY**(**pulseTimePercents**);**

xCoord **+=** xStep**;**

s**.**Points**.**Add**(**pt**);**

**}** // for

**}**

/// <summary>

/// обрабатываем ответ от контроллера

/// </summary>

/// <param name="dt">строка, которая пришла из порта</param>

**private** List**<**byte**>** COMAnswer **=** **new** List**<**byte**>();**

**private** void ProcessPortAnswer**(**byte**[]** dt**)**

**{**

EnsureCloseConnectionForm**();** // закрываем форму коннекта, если она ещё не закрыта

**switch** **(**answerBehaviour**)**

**{**

**case** AnswerBehaviour**.**Normal**:**

**{**

// нормальный режим работы

COMAnswer**.**AddRange**(**dt**);**

**while(true)**

**{**

int idx **=** Array**.**IndexOf**(**COMAnswer**.**ToArray**(),** **(**byte**)**'\n'**);**

**if** **(**idx **!=** **-**1**)**

**{**

string line **=** System**.**Text**.**Encoding**.**UTF8**.**GetString**(**COMAnswer**.**ToArray**(),** 0**,** idx**);**

COMAnswer**.**RemoveRange**(**0**,** idx**+**1**);**

line **=** line**.**Trim**();**

ProcessAnswerLine**(**line**);**

**}**

**else**

**break;**

**}**

**}**

**break;**

**case** AnswerBehaviour**.**SDCommandFILE**:**

**{**

fileReadedBytes **+=** dt**.**Length**;**

int percentsReading **=** **(**fileReadedBytes **\*** 100**)** **/** **(**requestedFileSize **==** 0 **?** 1 **:** requestedFileSize**);**

**if** **(**percentsReading **>** 100**)**

percentsReading **=** 100**;**

fileDownloadProgressFunction**?.**Invoke**(**percentsReading**,** dt**.**Length**);**

// вычитываем файл с SD. Признаком окончания файла служат байты [END]\r\n

**for** **(**int i **=** 0**;** i **<** dt**.**Length**;** i**++)**

**{**

**this.**SDQueryAnswer**.**Add**(**dt**[**i**]);**

// System.Diagnostics.Debug.Write((char) dt[i]);

**}**

**if(**SDQueryAnswer**.**Count **>** 6**)**

**{**

// уже можно проверять на окончание пакета

string endOfFile **=** ""**;**

**for(**int i**=** SDQueryAnswer**.**Count**-**7**;**i**<** SDQueryAnswer**.**Count**;**i**++)**

**{**

endOfFile **+=** **(**char**)**SDQueryAnswer**[**i**];**

**}** // for

**if(**endOfFile **==** "[END]\r\n"**)**

**{**

SDQueryAnswer**.**RemoveRange**(**SDQueryAnswer**.**Count **-** 7**,** 7**);**

//DealWithFileContent(SDQueryAnswer);

fileDataParseFunction**?.**Invoke**(**SDQueryAnswer**);**

SDQueryAnswer**.**Clear**();**

**this.**answerBehaviour **=** AnswerBehaviour**.**Normal**;**

**this.**currentCommand**.**ParseFunction **=** **null;** // освобождаем

**this.**currentCommand**.**CommandToSend **=** ""**;**

**}**

**}** // if

**}**

**break;**

**case** AnswerBehaviour**.**SDCommandLS**:**

**{**

// опрашиваем SD, команда LS

**for** **(**int i **=** 0**;** i **<** dt**.**Length**;** i**++)**

**this.**SDQueryAnswer**.**Add**(**dt**[**i**]);**

// тут разбиваем по '\n', т.к. ответ на LS - всегда текстовый

**while(true)**

**{**

int newLineIdx **=** SDQueryAnswer**.**FindIndex**(**x **=>** x **==** '\n'**);**

**if** **(**newLineIdx **!=** **-**1**)**

**{**

// нашли перевод строки

string lsLine **=** ""**;**

**for** **(**int k **=** 0**;** k **<** newLineIdx**;** k**++)**

**{**

lsLine **+=** **(**char**)**SDQueryAnswer**[**k**];**

**}**

SDQueryAnswer**.**RemoveRange**(**0**,** newLineIdx **+** 1**);**

lsLine **=** lsLine**.**Trim**();**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"<= COM: " **+** lsLine**);**

**if** **(**lsLine **==** "[END]"**)** // закончили список!!!

**{**

**this.**answerBehaviour **=** AnswerBehaviour**.**Normal**;**

**this.**currentCommand**.**ParseFunction **=** **null;** // освобождаем

**this.**currentCommand**.**CommandToSend **=** ""**;**

lsParseFunction**?.**Invoke**();**

**break;**

**}**

**else**

**{**

// продолжаем список

lsRecordFunction**?.**Invoke**(**lsLine**);**

**}**

**}** // if

**else**

**break;**

**}**

**}**

**break;**

**}**

**}**

**private** void AddRecordToSDList**(**string line**,** TreeNode parent **=** **null)**

**{**

TreeNodeCollection nodes **=** **this.**treeViewSD**.**Nodes**;**

**if** **(**parent **!=** **null)**

**{**

nodes **=** parent**.**Nodes**;**

SDNodeTagHelper existingTag **=** **(**SDNodeTagHelper**)**parent**.**Tag**;**

parent**.**Tag **=** **new** SDNodeTagHelper**(**SDNodeTags**.**TagFolderNode**,**existingTag**.**FileName**,** existingTag**.**IsDirectory**);** // говорим, что мы вычитали это дело

// и удаляем заглушку...

**for(**int i**=**0**;**i**<**parent**.**Nodes**.**Count**;**i**++)**

**{**

TreeNode child **=** parent**.**Nodes**[**i**];**

SDNodeTagHelper tg **=** **(**SDNodeTagHelper**)**child**.**Tag**;**

**if(**tg**.**Tag **==** SDNodeTags**.**TagDummyNode**)**

**{**

child**.**Remove**();**

**break;**

**}**

**}**

**}**

bool isDir **=** **false;**

int dirIdx **=** line**.**IndexOf**(**"<DIR>"**);**

**if(**dirIdx **!=** **-**1**)**

**{**

isDir **=** **true;**

line **=** line**.**Substring**(**0**,** dirIdx**).**Trim**();**

**}**

TreeNode node **=** nodes**.**Add**(**getTextFromFileName**(**line**));**

**if(**isDir**)**

**{**

node**.**ImageIndex **=** 0**;**

node**.**SelectedImageIndex **=** 0**;**

TreeNode dummy **=** node**.**Nodes**.**Add**(**"вычитываем...."**);**

dummy**.**Tag **=** **new** SDNodeTagHelper**(**SDNodeTags**.**TagDummyNode**,**""**,false);** // этот узел потом удалим, при перечитывании

dummy**.**ImageIndex **=** **-**1**;**

node**.**Tag **=** **new** SDNodeTagHelper**(**SDNodeTags**.**TagFolderUninitedNode**,**line**,** isDir**);** // говорим, что мы не перечитали содержимое папки ещё

**}**

**else**

**{**

node**.**ImageIndex **=** 2**;**

node**.**SelectedImageIndex **=** node**.**ImageIndex**;**

node**.**Tag **=** **new** SDNodeTagHelper**(**SDNodeTags**.**TagFileNode**,**line**,** isDir**);**

**}**

**}**

/// <summary>

/// Обработчик события "Пришла строка из транспортного уровня"

/// </summary>

/// <param name="a"></param>

**public** **delegate** void DataParseFunction**(**Answer a**);**

**public** **delegate** void BeforeSendFunction**();**

**public** **delegate** void AfterSendFunction**();**

**public** **delegate** void LSDoneFunction**();**

**public** **delegate** void LSRecordFunction**(**string line**);**

**public** **delegate** void FileDownloadProgressFunction**(**int percentsCompleted**,** int bytesReceived**);**

**public** **delegate** void FileDataParseFunction**(**List**<**byte**>** content**);**

/// <summary>

/// Структура команды на обработку

/// </summary>

**private** struct QueuedCommand

**{**

**public** string CommandToSend**;**

**public** DataParseFunction ParseFunction**;**

**public** BeforeSendFunction BeforeSend**;**

**public** AfterSendFunction AfterSend**;**

**};**

/// <summary>

/// Помещаем команду в очередь на обработку

/// </summary>

/// <param name="cmd">Текстовая команда для контроллера</param>

/// <param name="act">К какому действию команда привязана</param>

/// <param name="func">Указатель на функцию-обработчик ответа от контроллера</param>

**public** void PushCommandToQueue**(**string cmd**,** DataParseFunction func**,** BeforeSendFunction before **=** **null,** AfterSendFunction after **=** **null)**

**{**

QueuedCommand q **=** **new** QueuedCommand**();**

q**.**CommandToSend **=** cmd**;**

q**.**ParseFunction **=** func**;**

q**.**BeforeSend **=** before**;**

q**.**AfterSend **=** after**;**

**if** **(!**commandsQueue**.**Contains**(**q**))**

commandsQueue**.**Enqueue**(**q**);**

**}**

/// <summary>

/// Возвращаем команду из очереди

/// </summary>

/// <param name="outCmd">Команда, которая получена из очереди</param>

/// <returns>Возвращаем false, если команд в очереди нет, иначе - true</returns>

**private** bool GetCommandFromQeue**(ref** QueuedCommand outCmd**)**

**{**

**if** **(**commandsQueue**.**Count **<** 1**)**

**return** **false;**

outCmd **=** commandsQueue**.**Dequeue**();**

**return** **true;**

**}**

/// <summary>

/// Очередь команд

/// </summary>

**private** Queue**<**QueuedCommand**>** commandsQueue **=** **new** Queue**<**QueuedCommand**>();**

/// <summary>

/// текущая команда на обработку

/// </summary>

**private** QueuedCommand currentCommand **=** **new** QueuedCommand**();**

**private** Color currentLogItemColor **=** SystemColors**.**ControlLight**;**

/// <summary>

/// добавляем строку в лог

/// </summary>

/// <param name="line">строка для добавления в лог</param>

**private** void AddToLog**(**string line**,** bool shouldAddCommandName**)**

**{**

line **=** line**.**Trim**();**

**if** **(**line**.**Length **<** 1**)**

**return;**

lvLog**.**BeginUpdate**();**

int cnt **=** **this.**lvLog**.**Items**.**Count**;**

**if** **(**cnt **>** 100**)**

**{**

**this.**lvLog**.**Items**.**RemoveAt**(**0**);**

**}**

ListViewItem li **=** **this.**lvLog**.**Items**.**Add**(**shouldAddCommandName **?** currentCommand**.**CommandToSend **:** ""**);**

li**.**ImageIndex **=** 0**;**

**if** **(**currentLogItemColor **==** SystemColors**.**Window**)**

currentLogItemColor **=** SystemColors**.**ControlLight**;**

**else**

currentLogItemColor **=** SystemColors**.**Window**;**

string prefix **=** "=> "**;**

**if** **(!**shouldAddCommandName**)**

prefix **=** "<= "**;**

li**.**BackColor **=** currentLogItemColor**;**

li**.**SubItems**.**Add**(**prefix **+** line**);**

li**.**EnsureVisible**();**

lvLog**.**EndUpdate**();**

**}**

/// <summary>

/// Обрабатываем результат соединения с портом

/// </summary>

/// <param name="succ"></param>

/// <param name="message"></param>

**private** void OnCOMConnect**(**bool succ**,** string message**)**

**{**

**if** **(**InvokeRequired**)**

**{**

Invoke**((**MethodInvoker**)delegate** **{** OnCOMConnect**(**succ**,** message**);** **});**

**return;**

**}**

lastConnected **=** succ**;**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"TRANSPORT CONNECT EVENT!"**);**

// обнуляем текущее состояние при переконнекте

//this.currentCommand.ActionToSet = Actions.None;

EnsureCloseConnectionForm**();**

**if** **(**succ**)**

**{**

**this.**btnConnect**.**ImageIndex **=** 1**;**

**this.**btnConnect**.**Text **=** "Соединено"**;**

InitAfterConnect**(true);**

**}**

**else**

**{**

InitAfterConnect**(false);**

MessageBox**.**Show**(**"Не удалось соединиться с устройством!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

/\*

if (succ && connForm != null)

{

connForm.lblCurrentAction.Text = "Ждём данные из порта...";

}

\*/

System**.**Diagnostics**.**Debug**.**WriteLine**(**"TRANSPORT CONNECT EVENT DONE!"**);**

**}**

**private** bool ethalonsRequested **=** **false;**

/// <summary>

/// Инициализируем необходимое после успешного коннекта

/// </summary>

**private** void InitAfterConnect**(**bool isConnected**)**

**{**

Config**.**Instance**.**Clear**();**

ethalonsRequested **=** **false;**

inUploadFileToController **=** **false;**

nudDelta1**.**Value **=** 0**;**

//DEPRECATED: nudDelta2.Value = 0;

//DEPRECATED: nudDelta3.Value = 0;

nudMotoresourceCurrent1**.**Value **=** 0**;**

//DEPRECATED: nudMotoresourceCurrent2.Value = 0;

//DEPRECATED: nudMotoresourceCurrent3.Value = 0;

nudMotoresourceMax1**.**Value **=** 0**;**

//DEPRECATED: nudMotoresourceMax2.Value = 0;

//DEPRECATED: nudMotoresourceMax3.Value = 0;

nudPulses1**.**Value **=** 0**;**

//DEPRECATED: nudPulses2.Value = 0;

//DEPRECATED: nudPulses3.Value = 0;

nudACSDelay**.**Value **=** 0**;**

nudRelayDelay**.**Value **=** 0**;**

nudHighBorder**.**Value **=** 0**;**

nudLowBorder**.**Value **=** 0**;**

nudHighBorder**.**Value **=** 0**;**

nudLowBorder**.**Value **=** 0**;**

tbControllerTime**.**Text **=** "-"**;**

tbFirmwareVersion**.**Text **=** ""**;**

tbFREERAM**.**Text **=** ""**;**

//DEPRECATED: ResetInductiveSensors();

ResetVoltage**();**

ClearEthalons**();**

btnViewSDFile**.**Enabled **=** treeViewSD**.**SelectedNode **!=** **null;**

btnDeleteSDFile**.**Enabled **=** btnViewSDFile**.**Enabled**;**

**if** **(**isConnected**)**

**{**

**this.**btnConnect**.**ImageIndex **=** 1**;**

**this.**btnConnect**.**Text **=** "Соединено"**;**

connectStatusMessage**.**Text **=** "Соединено."**;**

**}**

**else**

**{**

**this.**btnConnect**.**Text **=** "Соединить"**;**

**this.**btnConnect**.**ImageIndex **=** 0**;** // коннект оборвался

connectStatusMessage**.**Text **=** "Нет соединения."**;**

**}**

dateTimeFromControllerReceived **=** **false;**

inSetDateTimeToController **=** **true;**

inSetMotoresourceCurrentToController **=** **true;**

inSetMotoresourceMaxToController **=** **true;**

inSetPulsesToController **=** **true;**

inSetBordersToController **=** **true;**

inSetRelayDelayToController **=** **true;**

inSetDeltaToController **=** **true;**

uuidRequested **=** **false;**

// очищаем очередь

commandsQueue**.**Clear**();**

**this.**currentCommand**.**ParseFunction **=** **null;**

**this.**currentCommand**.**CommandToSend **=** ""**;**

**this.**coreBootFound **=** **false;**

**if** **(**isConnected**)**

**{**

InitTreeView**();**

// добавляем нужные команды для обработки сразу после коннекта

PushCommandToQueue**(**GET\_PREFIX **+** "UUID" **+** PARAM\_DELIMITER **+** GenerateUUID**(),** ParseAskUUID**);**

PushCommandToQueue**(**GET\_PREFIX **+** "DATETIME"**,** ParseAskDatetime**);**

PushCommandToQueue**(**GET\_PREFIX **+** "FREERAM"**,** ParseAskFreeram**);**

PushCommandToQueue**(**GET\_PREFIX **+** "RES\_CUR"**,** ParseAskMotoresurceCurrent**,** BeforeAskMotoresourceCurrent**);**

PushCommandToQueue**(**GET\_PREFIX **+** "RES\_MAX"**,** ParseAskMotoresurceMax**,** BeforeAskMotoresourceMax**);**

PushCommandToQueue**(**GET\_PREFIX **+** "PULSES"**,** ParseAskPulses**,** BeforeAskPulses**);**

PushCommandToQueue**(**GET\_PREFIX **+** "DELTA"**,** ParseAskDelta**,** BeforeAskDelta**);**

PushCommandToQueue**(**GET\_PREFIX **+** "TBORDERS"**,** ParseAskBorders**,** BeforeAskBorders**);**

PushCommandToQueue**(**GET\_PREFIX **+** "RDELAY"**,** ParseAskRelayDelay**,** BeforeAskRelayDelay**);**

//DEPRECATED: GetInductiveSensors();

GetVoltage**();**

RequestEthalons**();**

RescanSD**();**

ShowMainSettings**();**

tmPeriodicCommandsTimer**.**Enabled **=** **true;**

**}**

**else**

**{**

ClearAllData**();**

ShowStartPanel**();**

tmPeriodicCommandsTimer**.**Enabled **=** **false;**

**}**

**}**

**private** void setConnectionStatusMessage**()**

**{**

string cguid **=** Config**.**Instance**.**ControllerGUID**;**

string savedName **=** Config**.**Instance**.**ControllerGUID**;**

**if** **(**ControllerNames**.**Instance**.**Names**.**ContainsKey**(**cguid**))**

savedName **=** ControllerNames**.**Instance**.**Names**[**cguid**];**

**if** **(**IsConnected**())**

**{**

connectStatusMessage**.**Text **=** "Соединено, контроллер " **+** savedName**;**

**}**

**else**

connectStatusMessage**.**Text **=** ""**;**

**}**

**private** void ParseAskUUID**(**Answer a**)**

**{**

uuidRequested **=** **true;**

**if** **(**a**.**IsOkAnswer**)**

**{**

string receivedGUID **=** a**.**Params**[**1**];**

Config**.**Instance**.**ControllerGUID **=** receivedGUID**;**

string savedName **=** receivedGUID**;**

**if** **(**ControllerNames**.**Instance**.**Names**.**ContainsKey**(**receivedGUID**))**

savedName **=** ControllerNames**.**Instance**.**Names**[**receivedGUID**];**

**else**

ControllerNames**.**Instance**.**Names**[**receivedGUID**]** **=** ""**;**

setConnectionStatusMessage**();**

**}**

**}**

**private** string GenerateUUID**()**

**{**

**return** Guid**.**NewGuid**().**ToString**(**"N"**);**

**}**

**private** void RequestEthalons**()**

**{**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE|ETL|ET0UP.ETL"**,** DummyAnswerReceiver**,** SetSDFileReadingFlagEthalon**);**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE|ETL|ET0DWN.ETL"**,** DummyAnswerReceiver**,** SetSDFileReadingFlagEthalon**);**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE|ETL|ET1UP.ETL"**,** DummyAnswerReceiver**,** SetSDFileReadingFlagEthalon**);**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE|ETL|ET1DWN.ETL"**,** DummyAnswerReceiver**,** SetSDFileReadingFlagEthalon**);**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE|ETL|ET2UP.ETL"**,** DummyAnswerReceiver**,** SetSDFileReadingFlagEthalon**);**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE|ETL|ET2DWN.ETL"**,** DummyAnswerReceiver**,** SetSDFileReadingFlagEthalon**);**

**}**

**private** void BeforeAskDelta**()**

**{**

**this.**inSetDeltaToController **=** **true;**

**}**

**private** void BeforeAskPulses**()**

**{**

**this.**inSetPulsesToController **=** **true;**

**}**

**private** void BeforeAskBorders**()**

**{**

**this.**inSetBordersToController **=** **true;**

**}**

**private** void BeforeAskRelayDelay**()**

**{**

**this.**inSetRelayDelayToController **=** **true;**

**}**

**private** void BeforeAskMotoresourceCurrent**()**

**{**

**this.**inSetMotoresourceCurrentToController **=** **true;**

**}**

**private** void BeforeAskMotoresourceMax**()**

**{**

**this.**inSetMotoresourceMaxToController **=** **true;**

**}**

**private** void ClearAllData**()**

**{**

**this.**treeView**.**Nodes**[**0**].**Nodes**.**Clear**();**

**this.**treeViewSD**.**Nodes**.**Clear**();**

**this.**ClearInterruptsList**(this.**logDataGrid**);**

dateTimeFromControllerReceived **=** **false;**

tbFirmwareVersion**.**Text **=** ""**;**

tbFREERAM**.**Text **=** ""**;**

**}**

**private** void ParseAskDelta**(**Answer a**)**

**{**

**this.**inSetDeltaToController **=** **false;**

**if** **(**a**.**IsOkAnswer**)**

**{**

**try** **{** Config**.**Instance**.**Delta1 **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);** **}**

**catch**

**{**

Config**.**Instance**.**Delta1 **=** 0**;**

**}**

**}**

**else**

**{**

Config**.**Instance**.**Delta1 **=** 0**;**

**}**

**try**

**{**

nudDelta1**.**Value **=** Config**.**Instance**.**Delta1**;**

**}**

**catch**

**{**

nudDelta1**.**Value **=** 0**;**

Config**.**Instance**.**Delta1 **=** 0**;**

**}**

**}**

**private** void ParseAskPulses**(**Answer a**)**

**{**

**this.**inSetPulsesToController **=** **false;**

**if** **(**a**.**IsOkAnswer**)**

**{**

**try** **{** Config**.**Instance**.**Pulses1 **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);** **}**

**catch** **{** Config**.**Instance**.**Pulses1 **=** 0**;** **}**

**}**

**else**

**{**

Config**.**Instance**.**Pulses1 **=** 0**;**

**}**

**try**

**{**

nudPulses1**.**Value **=** Config**.**Instance**.**Pulses1**;**

**}**

**catch**

**{**

nudPulses1**.**Value **=** 0**;**

Config**.**Instance**.**Pulses1 **=** 0**;**

**}**

**}**

**private** void ParseAskBorders**(**Answer a**)**

**{**

**this.**inSetBordersToController **=** **false;**

**if** **(**a**.**IsOkAnswer**)**

**{**

**try** **{** Config**.**Instance**.**LowBorder **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);** **}** **catch** **{** Config**.**Instance**.**LowBorder **=** 0**;** **}**

**try** **{** Config**.**Instance**.**HighBorder **=** Convert**.**ToInt32**(**a**.**Params**[**2**]);** **}** **catch** **{** Config**.**Instance**.**HighBorder **=** 0**;** **}**

**}**

**else**

**{**

Config**.**Instance**.**LowBorder **=** 0**;**

Config**.**Instance**.**HighBorder **=** 0**;**

**}**

**try**

**{**

nudLowBorder**.**Value **=** Config**.**Instance**.**LowBorder**;**

**}**

**catch**

**{**

nudLowBorder**.**Value **=** 0**;**

Config**.**Instance**.**LowBorder **=** 0**;**

**}**

**try**

**{**

nudHighBorder**.**Value **=** Config**.**Instance**.**HighBorder**;**

**}**

**catch**

**{**

nudHighBorder**.**Value **=** 0**;**

Config**.**Instance**.**HighBorder **=** 0**;**

**}**

**}**

**private** void ParseAskRelayDelay**(**Answer a**)**

**{**

**this.**inSetRelayDelayToController **=** **false;**

**if** **(**a**.**IsOkAnswer**)**

**{**

**try** **{** Config**.**Instance**.**RelayDelay **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);** **}** **catch** **{** Config**.**Instance**.**RelayDelay **=** 0**;** **}**

**try** **{** Config**.**Instance**.**ACSDelay **=** Convert**.**ToInt32**(**a**.**Params**[**2**]);** **}** **catch** **{** Config**.**Instance**.**ACSDelay **=** 0**;** **}**

**}**

**else**

**{**

Config**.**Instance**.**RelayDelay **=** 0**;**

Config**.**Instance**.**ACSDelay **=** 0**;**

**}**

**try**

**{**

nudRelayDelay**.**Value **=** Config**.**Instance**.**RelayDelay**;**

**}**

**catch**

**{**

nudRelayDelay**.**Value **=** 0**;**

Config**.**Instance**.**RelayDelay **=** 0**;**

**}**

**try**

**{**

nudACSDelay**.**Value **=** Config**.**Instance**.**ACSDelay**;**

**}**

**catch**

**{**

nudACSDelay**.**Value **=** 0**;**

Config**.**Instance**.**ACSDelay **=** 0**;**

**}**

**}**

**private** void ParseAskMotoresurceCurrent**(**Answer a**)**

**{**

**this.**inSetMotoresourceCurrentToController **=** **false;**

**if(**a**.**IsOkAnswer**)**

**{**

**try** **{** Config**.**Instance**.**MotoresourceCurrent1 **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);** **}** **catch** **{** Config**.**Instance**.**MotoresourceCurrent1 **=** 0**;** **}**

**}**

**else**

**{**

Config**.**Instance**.**MotoresourceCurrent1 **=** 0**;**

**}**

**try**

**{**

nudMotoresourceCurrent1**.**Value **=** Config**.**Instance**.**MotoresourceCurrent1**;**

**}**

**catch**

**{**

nudMotoresourceCurrent1**.**Value **=** 0**;**

Config**.**Instance**.**MotoresourceCurrent1 **=** 0**;**

**}**

UpdateMotoresourcePercents**();**

**}**

**private** void UpdateMotoresourcePercents**()**

**{**

**if** **(this.**inSetMotoresourceMaxToController **||** **this.**inSetMotoresourceCurrentToController**)**

**return;**

NumberFormatInfo nfi **=** **new** NumberFormatInfo**();**

nfi**.**NumberDecimalSeparator **=** "."**;**

float percents **=** **(**Config**.**Instance**.**MotoresourceCurrent1 **\*** 100.0f**)** **/** Config**.**Instance**.**MotoresourceMax1**;**

Color foreColor **=** Color**.**Green**;**

**if** **(**percents **>=** 90.0f**)**

foreColor **=** Color**.**Red**;**

lblMotoresourcePercents1**.**ForeColor **=** foreColor**;**

lblMotoresourcePercents1**.**Text **=** percents**.**ToString**(**"n1"**,** nfi**)** **+** "%"**;**

/\*

//DEPRECATED:

foreColor = Color.Green;

percents = (Config.Instance.MotoresourceCurrent2 \* 100.0f) / Config.Instance.MotoresourceMax2;

if (percents >= 90.0f)

foreColor = Color.Red;

lblMotoresourcePercents2.ForeColor = foreColor;

lblMotoresourcePercents2.Text = percents.ToString("n1", nfi) + "%";

foreColor = Color.Green;

percents = (Config.Instance.MotoresourceCurrent3 \* 100.0f) / Config.Instance.MotoresourceMax3;

if (percents >= 90.0f)

foreColor = Color.Red;

lblMotoresourcePercents3.ForeColor = foreColor;

lblMotoresourcePercents3.Text = percents.ToString("n1", nfi) + "%";

\*/

**}**

**private** void ParseAskMotoresurceMax**(**Answer a**)**

**{**

**this.**inSetMotoresourceMaxToController **=** **false;**

**if** **(**a**.**IsOkAnswer**)**

**{**

**try** **{** Config**.**Instance**.**MotoresourceMax1 **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);** **}** **catch** **{** Config**.**Instance**.**MotoresourceMax1 **=** 0**;** **}**

**}**

**else**

**{**

Config**.**Instance**.**MotoresourceMax1 **=** 0**;**

**}**

**try**

**{**

nudMotoresourceMax1**.**Value **=** Config**.**Instance**.**MotoresourceMax1**;**

**}**

**catch**

**{**

nudMotoresourceMax1**.**Value **=** 0**;**

Config**.**Instance**.**MotoresourceMax1 **=** 0**;**

**}**

**}**

**private** bool dateTimeFromControllerReceived **=** **false;**

**private** DateTime controllerDateTime **=** DateTime**.**MinValue**;**

**private** void ParseAskDatetime**(**Answer a**)**

**{**

**this.**inSetDateTimeToController **=** **false;**

**if** **(**a**.**IsOkAnswer**)**

**{**

**try**

**{**

//пришло время из контроллера

**this.**controllerDateTime **=** DateTime**.**ParseExact**(**a**.**Params**[**1**],** "dd.MM.yyyy HH:mm:ss"**,** **null);**

dateTimeFromControllerReceived **=** **true;**

**}**

**catch** **{** **}**

**}**

**else**

**{**

**}**

**}**

**private** void ParseAskFreeram**(**Answer a**)**

**{**

**if** **(**a**.**IsOkAnswer**)**

**{**

//пришли данные о свободной памяти

tbFREERAM**.**Text **=** a**.**Params**[**1**];**

**}**

**else**

**{**

**}**

**}**

**private** FeaturesSettings featuresSettings **=** **new** FeaturesSettings**();**

**private** void RecreateTreeView**()**

**{**

**this.**treeView**.**Nodes**[**0**].**Nodes**.**Clear**();**

TreeNode node **=** **this.**treeView**.**Nodes**[**0**].**Nodes**.**Add**(**"Основные настройки"**);**

node**.**Tag **=** TreeNodeType**.**MainSettingsNode**;**

node**.**ImageIndex **=** 5**;**

node**.**SelectedImageIndex **=** node**.**ImageIndex**;**

**if** **(**featuresSettings**.**SDAvailable**)**

**{**

TreeNode n **=** **this.**treeView**.**Nodes**[**0**].**Nodes**.**Add**(**"SD-карта"**);**

n**.**Tag **=** TreeNodeType**.**SDSettingsNode**;**

n**.**ImageIndex **=** 8**;**

n**.**SelectedImageIndex **=** n**.**ImageIndex**;**

**}**

**}**

**private** string ByteAsHexString**(**byte b**)**

**{**

**return** "0x" **+** b**.**ToString**(**"X2"**);**

**}**

**private** void ResizeLogColumns**()**

**{**

**this.**logColumn1**.**Width **=** **this.**lvLog**.**ClientRectangle**.**Width **/** 2 **-** SystemInformation**.**VerticalScrollBarWidth **/** 2 **-** 2**;**

**this.**logColumn2**.**Width **=** **this.**logColumn1**.**Width**;**

**}**

**private** void InitArchive**()**

**{**

// загружаем архив

string path **=** Application**.**StartupPath **+** "\\Archive\\"**;**

**try**

**{**

System**.**IO**.**Directory**.**CreateDirectory**(**path**);**

**}**

**catch** **{** **}**

List**<**string**>** dirs **=** **new** List**<**string**>(**System**.**IO**.**Directory**.**EnumerateDirectories**(**path**));**

**foreach(**string dir **in** dirs**)**

**{**

string dirNameOnly **=** dir**.**Substring**(**dir**.**LastIndexOf**(**"\\"**)** **+** 1**);**

AddItemToArchive**(**dirNameOnly**);**

**}**

**}**

**private** void MainForm\_Load**(object** sender**,** EventArgs e**)**

**{**

InitSubstitutions**();**

EnumSerialPorts**();**

ResizeLogColumns**();**

InitArchive**();**

**this.**toolStrip**.**ImageList **=** toolbarImages**;**

**this.**btnConnect**.**ImageIndex **=** 0**;**

**this.**btnSetDateTime**.**ImageIndex **=** 3**;**

**this.**btnAbout**.**ImageIndex **=** 5**;**

**this.**btnUploadEthalon**.**ImageIndex **=** 4**;**

**this.**btnDisconnect**.**ImageIndex **=** 6**;**

**this.**btnControllerName**.**ImageIndex **=** 8**;**

**this.**btnImportSettings**.**ImageIndex **=** 9**;**

**this.**btnRecordEthalonUp**.**ImageIndex **=** 10**;**

**this.**btnRecordEthalonDown**.**ImageIndex **=** 11**;**

plMainSettings**.**Dock **=** DockStyle**.**Fill**;**

**this.**plSDSettings**.**Dock **=** DockStyle**.**Fill**;**

**this.**logDataGrid**.**Dock **=** DockStyle**.**Fill**;**

**this.**plEthalonChart**.**Dock **=** DockStyle**.**Fill**;**

**this.**plEmptySDWorkspace**.**Dock **=** DockStyle**.**Fill**;**

**this.**plArchiveEthalonChart**.**Dock **=** DockStyle**.**Fill**;**

**this.**archiveLogDataGrid**.**Dock **=** DockStyle**.**Fill**;**

**this.**plEmptySDWorkspace**.**BringToFront**();**

//TODO: тут остальные панели !!!

ShowStartPanel**();**

Application**.**Idle **+=** **new** EventHandler**(**Application\_Idle**);**

**}**

**private** void InitTreeView**()**

**{**

RecreateTreeView**();**

**this.**treeView**.**Nodes**[**0**].**ExpandAll**();**

**}**

**private** bool uuidRequested **=** **false;**

**private** bool lastConnected **=** **false;**

/// <summary>

/// Обработчик простоя

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void Application\_Idle**(object** sender**,** EventArgs e**)**

**{**

bool bConnected **=** IsConnected**();**

**if(**lastConnected **&&** **!**bConnected**)**

**{**

lastConnected **=** **false;**

InitAfterConnect**(false);**

**}**

btnUploadEthalon**.**Enabled **=** bConnected **&&** **!**inUploadFileToController**;**

btnControllerName**.**Enabled **=** bConnected **&&** uuidRequested**;**

btnImportSettings**.**Enabled **=** bConnected**;**

//btnSaveEthalons.Enabled = ethalonsRequested;

btnSetDateTime**.**Enabled **=** bConnected **&&** **!**inSetDateTimeToController**;**

btnSetDateTime2**.**Enabled **=** bConnected **&&** **!**inSetDateTimeToController**;**

**this.**btnDisconnect**.**Enabled **=** bConnected **&&** currentTransport **!=** **null;**

**this.**btnSetMotoresourceCurrent**.**Enabled **=** bConnected **&&** **!**inSetMotoresourceCurrentToController**;**

**this.**btnSetMotoresourceMax**.**Enabled **=** bConnected **&&** **!**inSetMotoresourceMaxToController**;**

**this.**btnSetPulses**.**Enabled **=** bConnected **&&** **!**inSetPulsesToController**;**

**this.**btnSetDelta**.**Enabled **=** bConnected **&&** **!**inSetDeltaToController**;**

**this.**btnSetBorders**.**Enabled **=** bConnected **&&** **!**inSetBordersToController**;**

**this.**btnSetRelayDelay**.**Enabled **=** bConnected **&&** **!**inSetRelayDelayToController**;**

**this.**btnRecordEthalonUp**.**Enabled **=** bConnected **&&** **!**inSetEthalonRecordToController**;**

**this.**btnRecordEthalonDown**.**Enabled **=** bConnected **&&** **!**inSetEthalonRecordToController**;**

**if** **(!**bConnected**)** // порт закрыт

**{**

**if** **(this.**lastSelectedPort **!=** **null)**

**{**

**this.**lastSelectedPort**.**Checked **=** **false;**

**}**

**}**

**else**

**{**

**if** **(**lastSelectedPort **!=** **null)**

**if** **(!**lastSelectedPort**.**Checked**)**

lastSelectedPort**.**Checked **=** **true;**

**}**

**}**

/// <summary>

/// Перечисляем COM-порты

/// </summary>

**private** void EnumSerialPorts**()**

**{**

miPort**.**DropDownItems**.**Clear**();**

string**[]** ports **=** SerialPort**.**GetPortNames**();**

**foreach** **(**string port **in** ports**)**

**{**

ToolStripMenuItem ti **=** **new** ToolStripMenuItem**(**port**);**

ti**.**MergeIndex **=** 1**;**

ti**.**AutoSize **=** **true;**

ti**.**ImageScaling **=** ToolStripItemImageScaling**.**None**;**

ti**.**Tag **=** port**;**

ti**.**Click **+=** ConnectToSelectedComPort**;**

ti**.**CheckOnClick **=** **false;**

miPort**.**DropDownItems**.**Add**(**ti**);**

**}**

**}**

**private** void ConnectToSelectedComPort**(object** sender**,** EventArgs e**)**

**{**

ToolStripMenuItem mi **=** **(**ToolStripMenuItem**)**sender**;**

**if** **(**mi**.**Checked**)**

**return;**

Disconnect**();**

**if** **(this.**lastSelectedPort **!=** **null)**

**{**

**this.**lastSelectedPort**.**Checked **=** **false;**

**}**

**this.**lastSelectedPort **=** mi**;**

mi**.**Checked **=** **true;**

DoConnect**((**string**)**mi**.**Tag**,** **true,** **false);**

**}**

**private** int GetConnectionSpeed**()**

**{**

**foreach(**ToolStripMenuItem tmi **in** **this.**portSpeedToolStripMenuItem**.**DropDownItems**)**

**{**

**if(**tmi**.**Checked**)**

**{**

int speed **=** Convert**.**ToInt32**(**tmi**.**Text**);**

**return** speed**;**

**}**

**}**

**return** 115200**;**

**}**

/// <summary>

/// Начинаем коннектиться к порту

/// </summary>

/// <param name="port">имя порта</param>

**private** void DoConnect**(**string port**,** bool withHandshake**,** bool findDevice**)**

**{**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"START CONNECT...."**);**

dateTimeFromControllerReceived **=** **false;** // не получили ещё текущее время с контроллера

controllerDateTime **=** DateTime**.**MinValue**;** // устанавливаем минимальное значение даты

connForm **=** **new** ConnectForm**(false);**

connForm**.**SetMainFormAndPort**(this,** port**,** withHandshake**,** findDevice**);**

connForm**.**ShowDialog**();**

**}**

/// <summary>

/// Отсоединяемся

/// </summary>

**private** void Disconnect**()** // отсоединяемся от порта

**{**

**if** **(**currentTransport **!=** **null)**

currentTransport**.**Disconnect**();**

currentTransport **=** **null;**

**}**

/// <summary>

/// Проверяем, соединены ли мы с контроллером

/// </summary>

/// <returns></returns>

**public** bool IsConnected**()**

**{**

**if** **(**currentTransport **!=** **null)**

**return** currentTransport**.**Connected**();**

**return** **false;**

**}**

/// <summary>

/// Класс ответа от контроллера

/// </summary>

**public** class Answer

**{**

/// <summary>

/// флаг, что ответ положительный

/// </summary>

**public** bool IsOkAnswer**;**

/// <summary>

/// список параметров

/// </summary>

**public** string**[]** Params**;**

/// <summary>

/// сырые данные, полученные от контроллера

/// </summary>

**public** string RawData**;**

/// <summary>

/// очищает все переменные

/// </summary>

**private** void Clear**()**

**{**

IsOkAnswer **=** **false;**

Params **=** **null;**

RawData **=** ""**;**

**}**

/// <summary>

/// конструирует параметры из строки

/// </summary>

/// <param name="dt"></param>

**public** void Parse**(**string dt**)**

**{**

Clear**();**

RawData **=** dt**;**

int idx **=** dt**.**IndexOf**(**"OK="**);**

**if** **(**idx **!=** **-**1**)**

**{**

**this.**IsOkAnswer **=** **true;**

dt **=** dt**.**Substring**(**3**).**Trim**();**

**this.**Params **=** dt**.**Split**(**PARAM\_DELIMITER**);**

**}**

idx **=** dt**.**IndexOf**(**"ER="**);**

**if** **(**idx **!=** **-**1**)**

**{**

**this.**IsOkAnswer **=** **false;**

dt **=** dt**.**Substring**(**3**).**Trim**();**

**this.**Params **=** dt**.**Split**(**PARAM\_DELIMITER**);**

**}**

**}**

/// <summary>

/// кол-во параметров

/// </summary>

**public** int ParamsCount

**{**

get **{** **return** **this.**Params **==** **null** **?** 0 **:** **this.**Params**.**Length**;** **}**

**}**

/// <summary>

/// конструктор

/// </summary>

/// <param name="dt">строка для разбора</param>

**public** Answer**(**string dt**)**

**{**

**this.**Clear**();**

**this.**Parse**(**dt**);**

**}**

**}**

**private** const int everyNTimerTicksRequestLastTrig **=** 5**;**

**private** int timerTicksCounter **=** 0**;**

**private** void SetLastTrigReadingFlag**()**

**{**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandFILE**;**

**this.**fileDataParseFunction **=** **this.**ViewLastTrigData**;**

**this.**SDQueryAnswer**.**Clear**();**

**}**

**private** void LastTrigCallback**(**InterruptRecord rec**)**

**{**

// тут показываем форму последнего срабатывания

string stationID **=** Config**.**Instance**.**ControllerGUID**;**

string stationName **=** stationID**;**

**if** **(**ControllerNames**.**Instance**.**Names**.**ContainsKey**(**stationID**))**

stationName **=** ControllerNames**.**Instance**.**Names**[**stationID**];**

ShowChart**(**rec**,** stationID**,** stationName**,false);**

**}**

**private** void ViewLastTrigData**(**List**<**byte**>** content**)**

**{**

**if** **(**InvokeRequired**)**

**{**

Invoke**((**MethodInvoker**)delegate** **{** ViewLastTrigData**(**content**);** **});**

**return;**

**}**

**if** **(**content**.**Count **>** 0**)** // есть информация по срабатыванию!!!

**{**

ShowLogFile**(**content**,** **null,** ""**,** **false,** **null,** LastTrigCallback**,** **true);**

**}**

**}**

**private** void tmProcessCommandsTimer\_Tick**(object** sender**,** EventArgs e**)**

**{**

ProcessNextCommand**();**

**}**

**private** AnswerBehaviour answerBehaviour **=** AnswerBehaviour**.**Normal**;**

**private** void ProcessNextCommand**()**

**{**

**if** **(!**GrantToProcess**())**

**return;**

timerTicksCounter**++;**

**if(**timerTicksCounter **>=** everyNTimerTicksRequestLastTrig**)**

**{**

timerTicksCounter **=** 0**;**

PushCommandToQueue**(**GET\_PREFIX **+** "LASTTRIG"**,** DummyAnswerReceiver**,** SetLastTrigReadingFlag**);**

**}**

**if** **(!this.**GetCommandFromQeue**(ref** **this.**currentCommand**))**

**return;**

System**.**Diagnostics**.**Debug**.**Assert**(this.**currentTransport **!=** **null);**

**this.**answerBehaviour **=** AnswerBehaviour**.**Normal**;**

currentCommand**.**BeforeSend**?.**Invoke**();**

**this.**currentTransport**.**WriteLine**(**currentCommand**.**CommandToSend**);**

currentCommand**.**AfterSend**?.**Invoke**();**

**}**

**private** bool coreBootFound **=** **false;**

/// <summary>

/// Проверяем, можем ли мы работать

/// </summary>

/// <returns>возвращаем true, если работать можем</returns>

**private** bool GrantToProcess**()**

**{**

**if** **(!**coreBootFound**)**

**return** **false;**

**if** **(!this.**IsConnected**())** // нет коннекта

**{**

**return** **false;**

**}**

**if** **(this.**currentCommand**.**ParseFunction **!=** **null)** // чем-то заняты

**return** **false;**

**return** **true;**

**}**

**private** void tmGetSensorsData\_Tick**(object** sender**,** EventArgs e**)**

**{**

**if** **(!**GrantToProcess**())**

**return;**

PushCommandToQueue**(**GET\_PREFIX **+** "FREERAM"**,** ParseAskFreeram**);**

**}**

**private** void ShowStartPanel**()**

**{**

**this.**plStartPanel**.**Dock **=** DockStyle**.**Fill**;**

**this.**plStartPanel**.**BringToFront**();**

**}**

**private** void ShowMainSettings**()**

**{**

**this.**plMainSettings**.**BringToFront**();**

treeView**.**SelectedNode **=** treeView**.**Nodes**[**0**].**Nodes**[**0**];**

**}**

**private** void propertyGridSettings\_PropertyValueChanged**(object** s**,** PropertyValueChangedEventArgs e**)**

**{**

**}**

**private** void tmDateTime\_Tick**(object** sender**,** EventArgs e**)**

**{**

**if** **(!**dateTimeFromControllerReceived**)**

**{**

tbControllerTime**.**Text **=** "-"**;**

**return;**

**}**

**this.**controllerDateTime **=** **this.**controllerDateTime**.**AddMilliseconds**(**tmDateTime**.**Interval**);**

string dateTimeString **=** **this.**controllerDateTime**.**ToString**(**"dd.MM.yyyy HH:mm:ss"**);**

tbControllerTime**.**Text **=** dateTimeString**;**

**}**

**private** DateTime dateTimeToSet **=** DateTime**.**MinValue**;**

**private** bool inSetDateTimeToController **=** **false;**

**private** void btnSetDateTime\_Click**(object** sender**,** EventArgs e**)**

**{**

DialogResult dr **=** MessageBox**.**Show**(**"Установить время контроллера в локальное время компьютера?"**,** "Подтверждение"**,** MessageBoxButtons**.**OKCancel**,** MessageBoxIcon**.**Question**);**

**if** **(**dr **!=** System**.**Windows**.**Forms**.**DialogResult**.**OK**)**

**return;**

ShowWaitCursor**(true);**

dateTimeToSet **=** DateTime**.**Now**;**

inSetDateTimeToController **=** **true;**

String s **=** string**.**Format**(**"{0,0:D2}.{1,0:D2}.{2} {3,0:D2}:{4,0:D2}:{5,0:D2}"**,** dateTimeToSet**.**Day**,** dateTimeToSet**.**Month**,** dateTimeToSet**.**Year**,** dateTimeToSet**.**Hour**,** dateTimeToSet**.**Minute**,** dateTimeToSet**.**Second**);**

PushCommandToQueue**(**SET\_PREFIX **+** "DATETIME" **+** PARAM\_DELIMITER **+** s**,** ParseSetDatetime**);**

**}**

**private** void ParseSetDelta**(**Answer a**)**

**{**

inSetDeltaToController **=** **false;**

ShowWaitCursor**(false);**

**if** **(**a**.**IsOkAnswer**)**

**{**

Config**.**Instance**.**Delta1 **=** Convert**.**ToInt32**(**nudDelta1**.**Value**);**

MessageBox**.**Show**(**"Дельты обновлёны."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

nudDelta1**.**Value **=** Config**.**Instance**.**Delta1**;**

MessageBox**.**Show**(**"Ошибка обновления дельт!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void ParseSetPulses**(**Answer a**)**

**{**

inSetPulsesToController **=** **false;**

ShowWaitCursor**(false);**

**if** **(**a**.**IsOkAnswer**)**

**{**

Config**.**Instance**.**Pulses1 **=** Convert**.**ToInt32**(**nudPulses1**.**Value**);**

MessageBox**.**Show**(**"Импульсы обновлёны."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

nudPulses1**.**Value **=** Config**.**Instance**.**Pulses1**;**

MessageBox**.**Show**(**"Ошибка обновления импульсов!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void ParseSetBorders**(**Answer a**)**

**{**

inSetBordersToController **=** **false;**

ShowWaitCursor**(false);**

**if** **(**a**.**IsOkAnswer**)**

**{**

Config**.**Instance**.**LowBorder **=** Convert**.**ToInt32**(**nudLowBorder**.**Value**);**

Config**.**Instance**.**HighBorder **=** Convert**.**ToInt32**(**nudHighBorder**.**Value**);**

MessageBox**.**Show**(**"Пороги трансформатора обновлёны."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

nudLowBorder**.**Value **=** Config**.**Instance**.**LowBorder**;**

nudHighBorder**.**Value **=** Config**.**Instance**.**HighBorder**;**

MessageBox**.**Show**(**"Ошибка обновления порогов трансформатора!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void ParseSetRelayDelay**(**Answer a**)**

**{**

inSetRelayDelayToController **=** **false;**

ShowWaitCursor**(false);**

**if** **(**a**.**IsOkAnswer**)**

**{**

Config**.**Instance**.**RelayDelay **=** Convert**.**ToInt32**(**nudRelayDelay**.**Value**);**

Config**.**Instance**.**ACSDelay **=** Convert**.**ToInt32**(**nudACSDelay**.**Value**);**

MessageBox**.**Show**(**"Настройки обновлены."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

nudRelayDelay**.**Value **=** Config**.**Instance**.**RelayDelay**;**

nudACSDelay**.**Value **=** Config**.**Instance**.**ACSDelay**;**

MessageBox**.**Show**(**"Ошибка обновления настроек!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void ParseSetMotoresourceCurrent**(**Answer a**)**

**{**

inSetMotoresourceCurrentToController **=** **false;**

ShowWaitCursor**(false);**

**if** **(**a**.**IsOkAnswer**)**

**{**

Config**.**Instance**.**MotoresourceCurrent1 **=** Convert**.**ToInt32**(**nudMotoresourceCurrent1**.**Value**);**

MessageBox**.**Show**(**"Текущий моторесурс обновлён."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

nudMotoresourceCurrent1**.**Value **=** Config**.**Instance**.**MotoresourceCurrent1**;**

MessageBox**.**Show**(**"Ошибка обновления текущего моторесурса!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

UpdateMotoresourcePercents**();**

**}**

**private** void ParseSetMotoresourceMax**(**Answer a**)**

**{**

inSetMotoresourceMaxToController **=** **false;**

ShowWaitCursor**(false);**

**if** **(**a**.**IsOkAnswer**)**

**{**

Config**.**Instance**.**MotoresourceMax1 **=** Convert**.**ToInt32**(**nudMotoresourceMax1**.**Value**);**

MessageBox**.**Show**(**"Максимальный моторесурс обновлён."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

nudMotoresourceMax1**.**Value **=** Config**.**Instance**.**MotoresourceMax1**;**

MessageBox**.**Show**(**"Ошибка обновления максимального моторесурса!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

UpdateMotoresourcePercents**();**

**}**

**private** void ParseSetDatetime**(**Answer a**)**

**{**

inSetDateTimeToController **=** **false;**

ShowWaitCursor**(false);**

**if** **(**a**.**IsOkAnswer**)**

**{**

**this.**dateTimeFromControllerReceived **=** **true;**

**this.**controllerDateTime **=** dateTimeToSet**;**

MessageBox**.**Show**(**"Время контроллера обновлено."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

MessageBox**.**Show**(**"Ошибка установки времени на контроллере!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void MainForm\_Resize**(object** sender**,** EventArgs e**)**

**{**

ResizeLogColumns**();**

**}**

**private** void treeView\_BeforeCollapse**(object** sender**,** TreeViewCancelEventArgs e**)**

**{**

//e.Cancel = true;

**}**

**private** void ShowWaitCursor**(**bool show**)**

**{**

Cursor**.**Current **=** show **?** Cursors**.**WaitCursor **:** Cursors**.**Default**;**

Application**.**UseWaitCursor **=** show**;**

Application**.**DoEvents**();**

**}**

**private** void btnAbout\_Click**(object** sender**,** EventArgs e**)**

**{**

AboutForm ab **=** **new** AboutForm**();**

ab**.**ShowDialog**();**

**}**

**private** TreeNode currentSDParentNode **=** **null;**

**private** void RescanSD**()**

**{**

btnListSDFiles**.**Enabled **=** **false;**

PushCommandToQueue**(**GET\_PREFIX **+** "LS"**,** DummyAnswerReceiver**,** SetSDReadingFlag**);**

**}**

**private** void btnListSDFiles\_Click**(object** sender**,** EventArgs e**)**

**{**

RescanSD**();**

**}**

**public** void DummyAnswerReceiver**(**Answer a**)**

**{**

**}**

**private** LSDoneFunction lsParseFunction **=** **null;**

**private** LSRecordFunction lsRecordFunction **=** **null;**

**private** FileDownloadProgressFunction fileDownloadProgressFunction **=** **null;**

**private** FileDataParseFunction fileDataParseFunction **=** **null;**

**private** void ParseSDLSRecord**(**string line**)**

**{**

AddToLog**(**line**,** **false);**

AddRecordToSDList**(**line**,** currentSDParentNode**);**

**}**

**private** void ParseSDLSDone**()**

**{**

**this.**btnListSDFiles**.**Enabled **=** **true;**

**if** **(**currentSDParentNode **!=** **null)**

**{**

// и удаляем заглушку...

**for** **(**int i **=** 0**;** i **<** currentSDParentNode**.**Nodes**.**Count**;** i**++)**

**{**

TreeNode child **=** currentSDParentNode**.**Nodes**[**i**];**

SDNodeTagHelper tg **=** **(**SDNodeTagHelper**)**child**.**Tag**;**

**if** **(**tg**.**Tag **==** SDNodeTags**.**TagDummyNode**)**

**{**

child**.**Remove**();**

**break;**

**}**

**}**

**}** // if

**}**

**private** void SetSDReadingFlag**()**

**{**

lsParseFunction **=** ParseSDLSDone**;**

lsRecordFunction **=** ParseSDLSRecord**;**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandLS**;**

**this.**currentSDParentNode **=** **null;**

**this.**SDQueryAnswer**.**Clear**();**

**this.**treeViewSD**.**Nodes**.**Clear**();**

**}**

**private** void SetSDFolderReadingFlag**()**

**{**

lsParseFunction **=** ParseSDLSDone**;**

lsRecordFunction **=** ParseSDLSRecord**;**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandLS**;**

**this.**currentSDParentNode **=** tempSDParentNode**;**

**this.**SDQueryAnswer**.**Clear**();**

**}**

**private** Dictionary**<**string**,** string**>** fileNamesSubstitutions **=** **new** Dictionary**<**string**,** string**>();**

**private** void InitSubstitutions**()**

**{**

fileNamesSubstitutions**.**Clear**();**

fileNamesSubstitutions**.**Add**(**"LOG"**,** "Лог-файлы"**);**

fileNamesSubstitutions**.**Add**(**"ETL"**,** "Эталоны"**);**

fileNamesSubstitutions**.**Add**(**"ET0UP.ETL"**,** "Канал №1, движение вверх"**);**

fileNamesSubstitutions**.**Add**(**"ET0DWN.ETL"**,** "Канал №1, движение вниз"**);**

fileNamesSubstitutions**.**Add**(**"ET1UP.ETL"**,** "Канал №2, движение вверх"**);**

fileNamesSubstitutions**.**Add**(**"ET1DWN.ETL"**,** "Канал №2, движение вниз"**);**

fileNamesSubstitutions**.**Add**(**"ET2UP.ETL"**,** "Канал №3, движение вверх"**);**

fileNamesSubstitutions**.**Add**(**"ET2DWN.ETL"**,** "Канал №3, движение вниз"**);**

**}**

string getFileNameFromText**(**string text**)**

**{**

**if(**fileNamesSubstitutions**.**ContainsValue**(**text**))**

**{**

string key **=** fileNamesSubstitutions**.**FirstOrDefault**(**x **=>** x**.**Value **==** text**).**Key**;**

**return** key**;**

**}**

**return** text**;**

**}**

string getTextFromFileName**(**string fName**)**

**{**

string fileName **=** fName**.**ToUpper**();**

**if(**fileNamesSubstitutions**.**ContainsKey**(**fileName**))**

**{**

**return** fileNamesSubstitutions**[**fileName**];**

**}**

**return** fileName**;**

**}**

TreeNode tempSDParentNode **=** **null;**

**private** void treeViewSD\_BeforeExpand**(object** sender**,** TreeViewCancelEventArgs e**)**

**{**

TreeNode wantedExpand **=** e**.**Node**;**

SDNodeTagHelper tg **=** **(**SDNodeTagHelper**)**wantedExpand**.**Tag**;**

**if** **(**tg**.**Tag **!=** SDNodeTags**.**TagFolderUninitedNode**)** // уже проинициализировали

**return;**

string folderName **=** tg**.**FileName**;**//getFileNameFromText(wantedExpand.Text);

TreeNode parent **=** wantedExpand**.**Parent**;**

**while** **(**parent **!=** **null)**

**{**

SDNodeTagHelper nt **=** **(**SDNodeTagHelper**)**parent**.**Tag**;**

folderName **=** /\*getFileNameFromText(parent.Text)\*/

nt**.**FileName **+** PARAM\_DELIMITER **+** folderName**;**

parent **=** parent**.**Parent**;**

**}**

tempSDParentNode **=** wantedExpand**;**

PushCommandToQueue**(**GET\_PREFIX **+** "LS" **+** PARAM\_DELIMITER **+** folderName**,** DummyAnswerReceiver**,** SetSDFolderReadingFlag**);**

**}**

**private** void SetSDFileReadingFlag**()**

**{**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandFILE**;**

**this.**fileDataParseFunction **=** **this.**ViewFile**;**

//this.fileDownloadFlags = FileDownloadFlags.View;

**this.**SDQueryAnswer**.**Clear**();**

ShowWaitCursor**(true);**

**}**

**private** int requestEthalonCounter **=** 0**;**

**private** void SetSDFileReadingFlagEthalon**()**

**{**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandFILE**;**

//this.fileDownloadFlags = FileDownloadFlags.DownloadEthalon;

**this.**fileDataParseFunction **=** **this.**SaveEthalonFile**;**

**this.**fileDownloadProgressFunction **=** **null;**

**this.**SDQueryAnswer**.**Clear**();**

**switch(**requestEthalonCounter**)**

**{**

**case** 0**:**

requestedFileName **=** "ET0UP.ETL"**;**

**break;**

**case** 1**:**

requestedFileName **=** "ET0DWN.ETL"**;**

**break;**

**case** 2**:**

requestedFileName **=** "ET1UP.ETL"**;**

**break;**

**case** 3**:**

requestedFileName **=** "ET1DWN.ETL"**;**

**break;**

**case** 4**:**

requestedFileName **=** "ET2UP.ETL"**;**

**break;**

**case** 5**:**

requestedFileName **=** "ET2DWN.ETL"**;**

**break;**

**}**

requestEthalonCounter**++;**

**}**

**private** string requestedFileName **=** ""**;**

**private** int requestedFileSize **=** 0**;**

**private** int fileReadedBytes **=** 0**;**

**private** void ShowDownloadPercents**(**int percents**,** int bytesReceived**)**

**{**

**this.**statusProgressBar**.**Value **=** percents**;**

**}**

**private** void RequestFile**(**TreeNode node**)**

**{**

**if** **(**node **==** **null)**

**return;**

**if** **(**node**.**Tag **==** **null)**

**return;**

SDNodeTagHelper tg **=** **(**SDNodeTagHelper**)**node**.**Tag**;**

**if** **(**tg**.**Tag **!=** SDNodeTags**.**TagFileNode**)**

**return;**

//fileDownloadFlags = FileDownloadFlags.View;

**this.**fileDataParseFunction **=** **this.**ViewFile**;**

**this.**fileDownloadProgressFunction **=** ShowDownloadPercents**;**

ShowWaitCursor**(true);**

string fullPathName **=** tg**.**FileName**;**//getFileNameFromText(selectedNode.Text);

TreeNode parent **=** node**.**Parent**;**

**while** **(**parent **!=** **null)**

**{**

SDNodeTagHelper nt **=** **(**SDNodeTagHelper**)**parent**.**Tag**;**

fullPathName **=** /\*getFileNameFromText(parent.Text)\*/ nt**.**FileName **+** PARAM\_DELIMITER **+** fullPathName**;**

parent **=** parent**.**Parent**;**

**}**

requestedFileName **=** fullPathName**;**

**this.**btnViewSDFile**.**Enabled **=** **false;**

**this.**btnDeleteSDFile**.**Enabled **=** **false;**

**this.**btnListSDFiles**.**Enabled **=** **false;**

PushCommandToQueue**(**GET\_PREFIX **+** "FILESIZE" **+** PARAM\_DELIMITER **+** fullPathName**,** ParseAskFileSize**);**

**}**

**private** void ParseAskFileSize**(**Answer a**)**

**{**

**if(!**a**.**IsOkAnswer**)**

**{**

**this.**btnViewSDFile**.**Enabled **=** treeViewSD**.**SelectedNode **!=** **null;**

**this.**btnDeleteSDFile**.**Enabled **=** treeViewSD**.**SelectedNode **!=** **null;**

**this.**btnListSDFiles**.**Enabled **=** **true;**

**return;**

**}**

requestedFileSize **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);**

fileReadedBytes **=** 0**;**

statusProgressMessage**.**Text **=** "Вычитываем файл \"" **+** requestedFileName **+** "\"..."**;**

statusProgressMessage**.**Visible **=** **true;**

statusProgressBar**.**Value **=** 0**;**

statusProgressBar**.**Visible **=** **true;**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE" **+** PARAM\_DELIMITER **+** requestedFileName**,** DummyAnswerReceiver**,** SetSDFileReadingFlag**);**

**}**

**private** ArchiveImportForm archiveImportForm **=** **null;**

**private** List**<**String**>** archiveLogsList **=** **new** List**<**string**>();**

**private** List**<**String**>** archiveEthalonsList **=** **new** List**<**string**>();**

**private** List**<**int**>** archiveLogsSizes **=** **new** List**<**int**>();**

**private** List**<**int**>** archiveEthalonsSizes **=** **new** List**<**int**>();**

**private** int archiveLogsIterator **=** 0**;**

**private** int archiveEthalonsIterator **=** 0**;**

**private** int waitForFileSizeCount **=** 0**;**

**private** int waitForFileSizeDone **=** 0**;**

**private** int archiveTotalFilesSize **=** 0**;**

**private** int archiveWaitForListDone **=** 0**;**

**private** int archiveListDoneCount **=** 0**;**

**public** void StartArchive**(**ArchiveImportForm fm**)**

**{**

archiveImportForm **=** fm**;**

archiveImportForm**.**DisableControls**();**

DoArchive**();**

**}**

**private** void StartListLogs**()**

**{**

lsParseFunction **=** ParseLogsDone**;**

lsRecordFunction **=** ParseLogsRecord**;**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandLS**;**

**this.**SDQueryAnswer**.**Clear**();**

**}**

**private** void ParseLogsDone**()**

**{**

archiveListDoneCount**++;**

**if** **(**archiveListDoneCount **>=** archiveWaitForListDone**)**

**{**

// получили список файлов, получаем их размеры

ArchiveRequestFileSizes**();**

**}**

**}**

**private** void ParseLogsRecord**(**string rec**)**

**{**

archiveLogsList**.**Add**(**rec**);**

**}**

**private** void StartListEthalons**()**

**{**

lsParseFunction **=** ParseEthalonsDone**;**

lsRecordFunction **=** ParseEthalonsRecord**;**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandLS**;**

**this.**SDQueryAnswer**.**Clear**();**

**}**

**private** void ParseEthalonsDone**()**

**{**

archiveListDoneCount**++;**

**if** **(**archiveListDoneCount **>=** archiveWaitForListDone**)**

**{**

// получили список файлов, получаем их размеры

ArchiveRequestFileSizes**();**

**}**

**}**

**private** void ParseEthalonsRecord**(**string rec**)**

**{**

archiveEthalonsList**.**Add**(**rec**);**

**}**

**private** void ArchiveRequestFileSizes**()**

**{**

archiveImportForm**.**lblMessage**.**Text **=** "Получаем размер файлов для архивирования..."**;**

archiveTotalFilesSize **=** 0**;**

**if(**archiveLogsList**.**Count **<** 1 **&&** archiveEthalonsList**.**Count **<** 1**)**

**{**

// оба списка пустых, надо импортировать только настройки

ArchiveImportSettings**();**

**return;**

**}**

// тут получаем список файлов

**if** **(**archiveEthalonsList**.**Count **>** 0**)**

waitForFileSizeCount**++;**

// requestedFileSize

**for** **(**int i**=**0**;**i**<**archiveEthalonsList**.**Count**;**i**++)**

**{**

string fullPathName **=** "ETL" **+** PARAM\_DELIMITER **+** archiveEthalonsList**[**i**];**

PushCommandToQueue**(**GET\_PREFIX **+** "FILESIZE" **+** PARAM\_DELIMITER **+** fullPathName**,** ArchiveParseEthalonFileSize**);**

**}**

**if** **(**archiveLogsList**.**Count **>** 0**)**

waitForFileSizeCount**++;**

// requestedFileSize

**for** **(**int i **=** 0**;** i **<** archiveLogsList**.**Count**;** i**++)**

**{**

string fullPathName **=** "LOG" **+** PARAM\_DELIMITER **+** archiveLogsList**[**i**];**

PushCommandToQueue**(**GET\_PREFIX **+** "FILESIZE" **+** PARAM\_DELIMITER **+** fullPathName**,** ArchiveParseLogsFileSize**);**

**}**

**}**

**private** void ArchiveParseEthalonFileSize**(**Answer a**)**

**{**

archiveEthalonsIterator**++;**

int fsize **=** 0**;**

**if(**a**.**IsOkAnswer**)**

**{**

fsize **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);**

archiveImportForm**.**pbProgress**.**Maximum **+=** fsize **+** 15**;**

**}**

archiveEthalonsSizes**.**Add**(**fsize**);**

**if(**archiveEthalonsIterator **>=** archiveEthalonsList**.**Count**)**

**{**

// закончили получение списка эталонов

waitForFileSizeDone**++;**

**if(**waitForFileSizeDone **>=** waitForFileSizeCount**)**

**{**

// закончили получение размеров для всех файлов, можно начинать скачивать

ArchiveDownloadFiles**();**

**}**

**}**

**}**

**private** void ArchiveParseLogsFileSize**(**Answer a**)**

**{**

archiveLogsIterator**++;**

int fsize **=** 0**;**

**if** **(**a**.**IsOkAnswer**)**

**{**

fsize **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);**

archiveImportForm**.**pbProgress**.**Maximum **+=** fsize**;**

**}**

archiveLogsSizes**.**Add**(**fsize**);**

**if** **(**archiveLogsIterator **>=** archiveLogsList**.**Count**)**

**{**

// закончили получение списка эталонов

waitForFileSizeDone**++;**

**if** **(**waitForFileSizeDone **>=** waitForFileSizeCount**)**

**{**

// закончили получение размеров для всех файлов, можно начинать скачивать

ArchiveDownloadFiles**();**

**}**

**}**

**}**

**private** int waitForArchiveDownloadListCount **=** 0**;**

**private** int waitForArchiveDownloadListDone **=** 0**;**

**private** void ArchiveDownloadFiles**()**

**{**

archiveImportForm**.**lblMessage**.**Text **=** "Начинаем скачивать файлы..."**;**

waitForArchiveDownloadListDone **=** 0**;**

waitForArchiveDownloadListCount **=** archiveLogsList**.**Count **+** archiveEthalonsList**.**Count**;**

// тут начинаем скачивать файлы

archiveEthalonsIterator **=** 0**;**

**for(**int i**=**0**;**i**<** archiveEthalonsList**.**Count**;**i**++)**

**{**

string filePath **=** "ETL" **+** PARAM\_DELIMITER **+** archiveEthalonsList**[**i**];**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE" **+** PARAM\_DELIMITER **+** filePath**,** DummyAnswerReceiver**,** ArchiveStartEthalonFileReading**);**

**}**

archiveLogsIterator **=** 0**;**

**for** **(**int i **=** 0**;** i **<** archiveLogsList**.**Count**;** i**++)**

**{**

string filePath **=** "LOG" **+** PARAM\_DELIMITER **+** archiveLogsList**[**i**];**

PushCommandToQueue**(**GET\_PREFIX **+** "FILE" **+** PARAM\_DELIMITER **+** filePath**,** DummyAnswerReceiver**,** ArchiveStartLogFileReading**);**

**}**

**}**

**private** void ArchiveStartEthalonFileReading**()**

**{**

archiveImportForm**.**lblMessage**.**Text **=** "Скачиваем файл \"" **+** archiveEthalonsList**[**archiveEthalonsIterator**]** **+** "\"..."**;**

requestedFileSize **=** archiveEthalonsSizes**[**archiveEthalonsIterator**];**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandFILE**;**

**this.**fileDataParseFunction **=** ArchiveEthalonFileReceived**;**

**this.**fileDownloadProgressFunction **=** ArchiveFileDownloadProgress**;**

**this.**SDQueryAnswer**.**Clear**();**

archiveEthalonsIterator**++;**

**}**

**private** void ArchiveStartLogFileReading**()**

**{**

archiveImportForm**.**lblMessage**.**Text **=** "Скачиваем файл \"" **+** archiveLogsList**[**archiveLogsIterator**]** **+** "\"..."**;**

requestedFileSize **=** archiveLogsSizes**[**archiveLogsIterator**];**

**this.**answerBehaviour **=** AnswerBehaviour**.**SDCommandFILE**;**

**this.**fileDataParseFunction **=** ArchiveLogFileReceived**;**

**this.**fileDownloadProgressFunction **=** ArchiveFileDownloadProgress**;**

**this.**SDQueryAnswer**.**Clear**();**

archiveLogsIterator**++;**

**}**

**private** void ArchiveImportSettings**()**

**{**

**if** **(**archiveImportForm**.**cbSettings**.**Checked**)**

**{**

// импортируем настройки

archiveImportForm**.**lblMessage**.**Text **=** "Импортируем настройки..."**;**

ArchiveSettings aSett **=** **new** ArchiveSettings**();**

aSett**.**ApplyFromConfig**();**

string filename **=** Application**.**StartupPath **+** "\\Archive\\" **+** Config**.**Instance**.**ControllerGUID **+** "\\Settings\\"**;**

**try**

**{**

System**.**IO**.**Directory**.**CreateDirectory**(**filename**);**

filename **+=** "settings.xml"**;**

aSett**.**Save**(**filename**);**

**}**

**catch** **{** **}**

**}**

//Тут заполнения дерева архива новой записью

AddItemToArchive**(**Config**.**Instance**.**ControllerGUID**);**

archiveImportForm**.**pbProgress**.**Value **=** archiveImportForm**.**pbProgress**.**Maximum**;**

archiveImportForm**.**lblMessage**.**Text **=** "Готово."**;**

MessageBox**.**Show**(**"Импорт успешно завершён!"**);**

archiveImportForm**.**Done**();**

archiveImportForm **=** **null;**

**}**

**private** void AddItemToArchive**(**string guid**)**

**{**

// ищем, есть ли такая запись. Если есть - обновляем, если нет - добавляем

TreeNode archiveNode **=** treeView**.**Nodes**[**1**];**

TreeNode existingNode **=** **null;**

ArchiveTreeRootItem atri **=** **null;**

**for** **(**int i**=**0**;**i**<** archiveNode**.**Nodes**.**Count**;**i**++)**

**{**

TreeNode child **=** archiveNode**.**Nodes**[**i**];**

**if(**child**.**Tag **is** ArchiveTreeRootItem**)**

**{**

atri **=** child**.**Tag **as** ArchiveTreeRootItem**;**

**if(**atri**.**GUID **==** guid**)**

**{**

existingNode **=** child**;**

**if** **(**ControllerNames**.**Instance**.**Names**.**ContainsKey**(**guid**))**

existingNode**.**Text **=** ControllerNames**.**Instance**.**Names**[**guid**];**

**else**

existingNode**.**Text **=** guid**;**

existingNode**.**Nodes**.**Clear**();**

**break;**

**}**

**}**

**}** // for

**if(**existingNode **==** **null)**

**{**

atri **=** **new** ArchiveTreeRootItem**(**guid**);**

existingNode **=** archiveNode**.**Nodes**.**Add**(**atri**.**ToString**());**

existingNode**.**Tag **=** atri**;**

existingNode**.**ImageIndex **=** 8**;**

existingNode**.**SelectedImageIndex **=** 8**;**

**}**

// тут добавляем дочерние ноды - эталоны и логи

TreeNode ethalonNode **=** existingNode**.**Nodes**.**Add**(**"Эталоны"**);**

ethalonNode**.**Tag **=** **new** ArchiveTreeEthalonItem**(**atri**);**

ethalonNode**.**ImageIndex **=** 14**;**

ethalonNode**.**SelectedImageIndex **=** 14**;**

FillArchiveEthalonsList**(**ethalonNode**,** ethalonNode**.**Tag **as** ArchiveTreeEthalonItem**);**

TreeNode logNode **=** existingNode**.**Nodes**.**Add**(**"Логи"**);**

logNode**.**Tag **=** **new** ArchiveTreeLogItem**(**atri**);**

logNode**.**ImageIndex **=** 13**;**

logNode**.**SelectedImageIndex **=** 13**;**

FillArchiveLogsList**(**logNode**,** logNode**.**Tag **as** ArchiveTreeLogItem**);**

**}**

**private** void FillArchiveEthalonsList**(**TreeNode n**,** ArchiveTreeEthalonItem eti**)**

**{**

n**.**Nodes**.**Clear**();**

string path **=** Application**.**StartupPath **+** "\\Archive\\" **+** eti**.**Parent**.**GUID **+** "\\ETL\\"**;**

**try**

**{**

System**.**IO**.**Directory**.**CreateDirectory**(**path**);**

**}**

**catch** **{** **}**

string**[]** files **=** System**.**IO**.**Directory**.**GetFiles**(**path**);**

**foreach(**string fullfilename **in** files**)**

**{**

string fileDisplayName **=** getTextFromFileName**(**System**.**IO**.**Path**.**GetFileName**(**fullfilename**));**

TreeNode child **=** n**.**Nodes**.**Add**(**fileDisplayName**);**

child**.**ImageIndex **=** 16**;**

child**.**SelectedImageIndex **=** 16**;**

ArchiveTreeEthalonItemRecord ateir **=** **new** ArchiveTreeEthalonItemRecord**(**eti**,** fullfilename**);**

child**.**Tag **=** ateir**;**

**}**

**}**

**private** void FillArchiveLogsList**(**TreeNode n**,** ArchiveTreeLogItem eti**)**

**{**

n**.**Nodes**.**Clear**();**

string path **=** Application**.**StartupPath **+** "\\Archive\\" **+** eti**.**Parent**.**GUID **+** "\\LOG\\"**;**

**try**

**{**

System**.**IO**.**Directory**.**CreateDirectory**(**path**);**

**}**

**catch** **{** **}**

string**[]** files **=** System**.**IO**.**Directory**.**GetFiles**(**path**);**

**foreach** **(**string file **in** files**)**

**{**

string fileDisplayName **=** System**.**IO**.**Path**.**GetFileName**(**file**);**

TreeNode child **=** n**.**Nodes**.**Add**(**fileDisplayName**);**

child**.**ImageIndex **=** 15**;**

child**.**SelectedImageIndex **=** 15**;**

ArchiveTreeLogItemRecord ateir **=** **new** ArchiveTreeLogItemRecord**(**eti**,** file**);**

child**.**Tag **=** ateir**;**

**}**

**}**

**private** void SaveArchiveFile**(**string fileName**,** string dirName**,** List**<**byte**>** content**)**

**{**

String path **=** Application**.**StartupPath **+** "\\Archive\\" **+** Config**.**Instance**.**ControllerGUID **+** "\\" **+** dirName **+** "\\"**;**

**try**

**{**

System**.**IO**.**Directory**.**CreateDirectory**(**path**);**

path **+=** fileName**;**

**try**

**{**

System**.**IO**.**BinaryWriter bw **=** **new** System**.**IO**.**BinaryWriter**(new** System**.**IO**.**FileStream**(**path**,** System**.**IO**.**FileMode**.**Create**));**

**for** **(**int i **=** 0**;** i **<** content**.**Count**;** i**++)**

bw**.**Write**(**content**[**i**]);**

bw**.**Close**();**

**}**

**catch**

**{**

**}**

**}**

**catch**

**{**

**}**

**}**

**private** void ArchiveEthalonFileReceived**(**List**<**byte**>** content**)**

**{**

// тут получены данные файла эталона

string fileName **=** archiveEthalonsList**[**archiveEthalonsIterator**-**1**];**

//тут сохраняем файл на диске

SaveArchiveFile**(**fileName**,** "ETL"**,** content**);**

waitForArchiveDownloadListDone**++;**

**if(**waitForArchiveDownloadListDone **>=** waitForArchiveDownloadListCount**)**

**{**

// получены все файлы, можно импортировать настройки

ArchiveImportSettings**();**

**}**

**}**

**private** void ArchiveLogFileReceived**(**List**<**byte**>** content**)**

**{**

// тут получены данные файла лога

string fileName **=** archiveLogsList**[**archiveLogsIterator **-** 1**];**

//тут сохраняем файл на диске

SaveArchiveFile**(**fileName**,** "LOG"**,** content**);**

waitForArchiveDownloadListDone**++;**

**if** **(**waitForArchiveDownloadListDone **>=** waitForArchiveDownloadListCount**)**

**{**

// получены все файлы, можно импортировать настройки

ArchiveImportSettings**();**

**}**

**}**

**private** void ArchiveFileDownloadProgress**(**int percents**,** int bytesReceived**)**

**{**

**try**

**{**

archiveImportForm**.**pbProgress**.**Value **+=** bytesReceived**;**

**}**

**catch**

**{**

**}**

**}**

**private** void DoArchive**()**

**{**

archiveLogsList**.**Clear**();**

archiveEthalonsList**.**Clear**();**

archiveLogsSizes**.**Clear**();**

archiveEthalonsSizes**.**Clear**();**

archiveLogsIterator **=** 0**;**

archiveEthalonsIterator **=** 0**;**

waitForFileSizeCount **=** 0**;**

waitForFileSizeDone **=** 0**;**

archiveTotalFilesSize **=** 0**;**

archiveWaitForListDone **=** 0**;**

archiveListDoneCount **=** 0**;**

ControllerNames**.**Instance**.**Names**[**Config**.**Instance**.**ControllerGUID**]** **=** archiveImportForm**.**tbControllerName**.**Text**.**Trim**();**

ControllerNames**.**Instance**.**Save**();**

archiveImportForm**.**lblMessage**.**Text **=** "Получаем список файлов для архивирования..."**;**

archiveImportForm**.**pbProgress**.**Maximum **=** 3**;**

archiveImportForm**.**pbProgress**.**Value **=** 1**;**

**if** **(**archiveImportForm**.**cbEthalons**.**Checked**)**

**{**

// запрошено архивирование эталонов

archiveWaitForListDone**++;**

PushCommandToQueue**(**GET\_PREFIX **+** "LS|ETL"**,** DummyAnswerReceiver**,** StartListEthalons**);**

**}**

**if** **(**archiveImportForm**.**cbLogs**.**Checked**)**

**{**

// запрошено архивирование логов

archiveWaitForListDone**++;**

PushCommandToQueue**(**GET\_PREFIX **+** "LS|LOG"**,** DummyAnswerReceiver**,** StartListLogs**);**

**}**

**if** **(**archiveListDoneCount **>=** archiveWaitForListDone**)**

**{**

// получили список файлов, получаем их размеры

ArchiveRequestFileSizes**();**

**}**

**}**

**private** void tmEnumComPorts\_Tick**(object** sender**,** EventArgs e**)**

**{**

string**[]** ports **=** SerialPort**.**GetPortNames**();**

// сначала удаляем те порты, которых нет в списке текущих

List**<**ToolStripMenuItem**>** toRemove **=** **new** List**<**ToolStripMenuItem**>();**

**foreach** **(**ToolStripMenuItem existing **in** miPort**.**DropDownItems**)**

**{**

bool found **=** **false;**

**foreach** **(**string port **in** ports**)**

**{**

**if(**port **==** existing**.**Text**)**

**{**

found **=** **true;**

**break;**

**}**

**}**

**if(!**found**)**

**{**

toRemove**.**Add**(**existing**);**

**}**

**}**

// теперь чистим

**for(**int i**=**0**;**i**<** toRemove**.**Count**;**i**++)**

**{**

miPort**.**DropDownItems**.**Remove**(**toRemove**[**i**]);**

**}**

**foreach** **(**string port **in** ports**)**

**{**

// ищем - есть ли такой порт уже?

bool found **=** **false;**

**foreach(**ToolStripMenuItem existing **in** miPort**.**DropDownItems**)**

**{**

**if(**existing**.**Text **==** port**)**

**{**

found **=** **true;**

**break;**

**}**

**}**

**if** **(**found**)**

**continue;**

ToolStripMenuItem ti **=** **new** ToolStripMenuItem**(**port**);**

ti**.**MergeIndex **=** 1**;**

ti**.**AutoSize **=** **true;**

ti**.**ImageScaling **=** ToolStripItemImageScaling**.**None**;**

ti**.**Tag **=** port**;**

ti**.**Click **+=** ConnectToSelectedComPort**;**

ti**.**CheckOnClick **=** **false;**

miPort**.**DropDownItems**.**Add**(**ti**);**

**}**

**}**

**private** void ChangePortSpeed**(object** sender**,** EventArgs e**)**

**{**

ToolStripMenuItem selItem **=** sender **as** ToolStripMenuItem**;**

**foreach(**ToolStripMenuItem tmi **in** **this.**portSpeedToolStripMenuItem**.**DropDownItems**)**

**{**

tmi**.**Checked **=** **false;**

**}**

selItem**.**Checked **=** **true;**

**}**

**private** bool inSetMotoresourceCurrentToController **=** **true;**

**private** void btnSetMotoresourceCurrent\_Click**(object** sender**,** EventArgs e**)**

**{**

inSetMotoresourceCurrentToController **=** **true;**

ShowWaitCursor**(true);**

string s **=** ""**;**

s **+=** Convert**.**ToString**(**nudMotoresourceCurrent1**.**Value**);**//DEPRECATED: + PARAM\_DELIMITER;

//DEPRECATED: s += Convert.ToString(nudMotoresourceCurrent2.Value) + PARAM\_DELIMITER;

//DEPRECATED: s += Convert.ToString(nudMotoresourceCurrent3.Value);

PushCommandToQueue**(**SET\_PREFIX **+** "RES\_CUR" **+** PARAM\_DELIMITER **+** s**,** ParseSetMotoresourceCurrent**);**

**}**

**private** bool inSetMotoresourceMaxToController **=** **true;**

**private** void btnSetMotoresourceMax\_Click**(object** sender**,** EventArgs e**)**

**{**

inSetMotoresourceMaxToController **=** **true;**

ShowWaitCursor**(true);**

string s **=** ""**;**

s **+=** Convert**.**ToString**(**nudMotoresourceMax1**.**Value**);**//DEPRECATED: + PARAM\_DELIMITER;

//DEPRECATED: s += Convert.ToString(nudMotoresourceMax2.Value) + PARAM\_DELIMITER;

//DEPRECATED: s += Convert.ToString(nudMotoresourceMax3.Value);

PushCommandToQueue**(**SET\_PREFIX **+** "RES\_MAX" **+** PARAM\_DELIMITER **+** s**,** ParseSetMotoresourceMax**);**

**}**

**private** bool inSetPulsesToController **=** **true;**

**private** void btnSetPulses\_Click**(object** sender**,** EventArgs e**)**

**{**

inSetPulsesToController **=** **true;**

ShowWaitCursor**(true);**

string s **=** ""**;**

s **+=** Convert**.**ToString**(**nudPulses1**.**Value**);**//DEPRECATED: + PARAM\_DELIMITER;

//DEPRECATED: s += Convert.ToString(nudPulses2.Value) + PARAM\_DELIMITER;

//DEPRECATED: s += Convert.ToString(nudPulses3.Value);

PushCommandToQueue**(**SET\_PREFIX **+** "PULSES" **+** PARAM\_DELIMITER **+** s**,** ParseSetPulses**);**

**}**

**private** bool inSetDeltaToController **=** **true;**

**private** void btnSetDelta\_Click**(object** sender**,** EventArgs e**)**

**{**

inSetDeltaToController **=** **true;**

ShowWaitCursor**(true);**

string s **=** ""**;**

s **+=** Convert**.**ToString**(**nudDelta1**.**Value**);**//DEPRECATED: + PARAM\_DELIMITER;

//DEPRECATED: s += Convert.ToString(nudDelta2.Value) + PARAM\_DELIMITER;

//DEPRECATED: s += Convert.ToString(nudDelta3.Value);

PushCommandToQueue**(**SET\_PREFIX **+** "DELTA" **+** PARAM\_DELIMITER **+** s**,** ParseSetDelta**);**

**}**

/\*

//DEPRECATED:

private void GetInductiveSensors()

{

PushCommandToQueue(GET\_PREFIX + "IND", ParseInductiveSensors);

}

\*/

**private** void GetVoltage**()**

**{**

PushCommandToQueue**(**GET\_PREFIX **+** "VDATA"**,** ParseVoltage**);**

**}**

**private** void GetMotoresourceCurrent**()**

**{**

**if** **(this.**inSetMotoresourceCurrentToController**)**

**return;**

PushCommandToQueue**(**GET\_PREFIX **+** "RES\_CUR"**,** ParseAskMotoresurceCurrent**,** BeforeAskMotoresourceCurrent**);**

**}**

**private** void tmInductiveTimer\_Tick**(object** sender**,** EventArgs e**)**

**{**

//DEPRECATED: GetInductiveSensors();

GetVoltage**();**

GetMotoresourceCurrent**();**

**}**

**private** void ResetVoltage**()**

**{**

lblVoltage1**.**BackColor **=** Color**.**LightGray**;**

lblVoltage1**.**Text **=** "-"**;**

lblVoltage2**.**BackColor **=** Color**.**LightGray**;**

lblVoltage2**.**Text **=** "-"**;**

lblVoltage3**.**BackColor **=** Color**.**LightGray**;**

lblVoltage3**.**Text **=** "-"**;**

**}**

**private** void ParseVoltage**(**Answer a**)**

**{**

**if** **(**a**.**IsOkAnswer**)**

**{**

int VOLTAGE\_THRESHOLD **=** 10**;**

NumberFormatInfo nfi **=** **new** NumberFormatInfo**();**

nfi**.**NumberDecimalSeparator **=** "."**;**

string end **=** "V"**;**

**try**

**{**

int vdata **=** Convert**.**ToInt32**(**a**.**Params**[**1**]);**

float threshold **=** **(**3.3f **/** 100**)** **\*** VOLTAGE\_THRESHOLD**;**

float lowBorder **=** 3.3f **-** threshold**;**

float highBorder **=** 3.3f **+** threshold**;**

float currentV **=** vdata **\*** **(**3.3f **/** 4096 **\*** 2**);**

**if** **(**currentV **>=** lowBorder **&&** currentV **<=** highBorder**)**

**{**

lblVoltage1**.**BackColor **=** Color**.**LightGreen**;**

**}**

**else**

**{**

lblVoltage1**.**BackColor **=** Color**.**LightSalmon**;**

**}**

lblVoltage1**.**Text **=** currentV**.**ToString**(**"n1"**,** nfi**)** **+** end**;**

**}**

**catch** **{** **}**

**try**

**{**

int vdata **=** Convert**.**ToInt32**(**a**.**Params**[**2**]);**

float threshold **=** **(**5.0f **/** 100**)** **\*** VOLTAGE\_THRESHOLD**;**

float lowBorder **=** 5.0f **-** threshold**;**

float highBorder **=** 5.0f **+** threshold**;**

float currentV **=** vdata **\*** **(**3.3f **/** 4096 **\*** 2**);**

**if** **(**currentV **>=** lowBorder **&&** currentV **<=** highBorder**)**

**{**

lblVoltage2**.**BackColor **=** Color**.**LightGreen**;**

**}**

**else**

**{**

lblVoltage2**.**BackColor **=** Color**.**LightSalmon**;**

**}**

lblVoltage2**.**Text **=** currentV**.**ToString**(**"n1"**,** nfi**)** **+** end**;**

**}**

**catch** **{** **}**

**try**

**{**

int vdata **=** Convert**.**ToInt32**(**a**.**Params**[**3**]);**

float threshold **=** **(**200.0f **/** 100**)** **\*** VOLTAGE\_THRESHOLD**;**

float lowBorder **=** 200.0f **-** threshold**;**

float highBorder **=** 200.0f **+** threshold**;**

float currentV **=** vdata **\*** **(**3.3f **/** 4096 **\*** 100**);**

**if** **(**currentV **>=** lowBorder **&&** currentV **<=** highBorder**)**

**{**

lblVoltage3**.**BackColor **=** Color**.**LightGreen**;**

**}**

**else**

**{**

lblVoltage3**.**BackColor **=** Color**.**LightSalmon**;**

**}**

lblVoltage3**.**Text **=** currentV**.**ToString**(**"n1"**,** nfi**)** **+** end**;**

**}**

**catch** **{** **}**

**}**

**else**

**{**

ResetVoltage**();**

**}**

**}**

**private** void MotoresourceCurrentValueChanged**(object** sender**,** EventArgs e**)**

**{**

UpdateMotoresourcePercents**();**

**}**

**private** void treeViewSD\_BeforeSelect**(object** sender**,** TreeViewCancelEventArgs e**)**

**{**

/\*

SDNodeTagHelper h = e.Node.Tag as SDNodeTagHelper;

if (h == null || h.IsDirectory)

e.Cancel = true;

\*/

**}**

**private** void treeViewSD\_AfterSelect**(object** sender**,** TreeViewEventArgs e**)**

**{**

btnDeleteSDFile**.**Enabled **=** **false;**

btnViewSDFile**.**Enabled **=** **false;**

**if** **(**treeViewSD**.**SelectedNode **==** **null)**

**{**

**return;**

**}**

SDNodeTagHelper h **=** e**.**Node**.**Tag **as** SDNodeTagHelper**;**

**if** **(**h **==** **null** **||** h**.**IsDirectory**)**

**return;**

btnDeleteSDFile**.**Enabled **=** **true;**

btnViewSDFile**.**Enabled **=** **true;**

**}**

**private** void btnDeleteSDFile\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**treeViewSD**.**SelectedNode **==** **null)**

**return;**

SDNodeTagHelper h **=** treeViewSD**.**SelectedNode**.**Tag **as** SDNodeTagHelper**;**

**if** **(**h **==** **null** **||** h**.**IsDirectory**)**

**return;**

string fileName **=** h**.**FileName**;**

string fAsk **=** treeViewSD**.**SelectedNode**.**Text**;**

**if** **(**fileName **!=** fAsk**)**

fAsk **+=** " (" **+** fileName **+** ")"**;**

**if** **(**MessageBox**.**Show**(**"Вы уверены, что хотите удалить файл \"" **+** fAsk **+** "\"?"**,** "Подтверждение"**,** MessageBoxButtons**.**YesNo**,** MessageBoxIcon**.**Question**)** **!=** DialogResult**.**Yes**)**

**return;**

TreeNode parent **=** treeViewSD**.**SelectedNode**.**Parent**;**

**while** **(**parent **!=** **null)**

**{**

SDNodeTagHelper nt **=** **(**SDNodeTagHelper**)**parent**.**Tag**;**

fileName **=** nt**.**FileName **+** PARAM\_DELIMITER **+** fileName**;**

parent **=** parent**.**Parent**;**

**}**

treeViewSD**.**SelectedNode**.**Remove**();**

PushCommandToQueue**(**SET\_PREFIX **+** "DELFILE" **+** PARAM\_DELIMITER **+** fileName**,** DummyAnswerReceiver**);**

**}**

**private** void btnViewSDFile\_Click**(object** sender**,** EventArgs e**)**

**{**

RequestFile**(**treeViewSD**.**SelectedNode**);**

**}**

**private** void treeViewSD\_MouseDoubleClick**(object** sender**,** MouseEventArgs e**)**

**{**

RequestFile**(**treeViewSD**.**SelectedNode**);**

**}**

**private** void ShowChart**(**InterruptRecord record**,** string stationID**,** string stationName**,** bool modal**)**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**record **!=** **null);**

ViewChartForm vcf **=** **new** ViewChartForm**(**record**,** stationID**,** stationName**);**

vcf**.**setDefaultFileName**(**record**.**InterruptInfo**.**InterruptTime**.**ToString**(**"yyyy-MM-dd HH.mm"**));**

vcf**.**lblCaption**.**Text **=** "Срабатывание от " **+** record**.**InterruptInfo**.**InterruptTime**.**ToString**(**"dd.MM.yyyy HH:mm:ss"**);**

System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**Series ethalonSerie **=** vcf**.**chart**.**Series**[**0**];**

ethalonSerie**.**Points**.**Clear**();**

System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**Series interruptSerie **=** vcf**.**chart**.**Series**[**1**];**

interruptSerie**.**Points**.**Clear**();**

int xStep **=** 1**;**

List**<**int**>** timeList **=** record**.**InterruptData**;**

// получаем максимальное время импульса - это будет 100% по оси Y

int maxPulseTime **=** 0**;**

**for** **(**int i **=** 1**;** i **<** timeList**.**Count**;** i**++)**

**{**

maxPulseTime **=** Math**.**Max**(**maxPulseTime**,** **(**timeList**[**i**]** **-** timeList**[**i **-** 1**]));**

**}**

int endStop **=** timeList**.**Count**;**

**for** **(**int i **=** 1**;** i **<** record**.**EthalonData**.**Count**;** i**++)**

**{**

maxPulseTime **=** Math**.**Max**(**maxPulseTime**,** **(**record**.**EthalonData**[**i**]** **-** record**.**EthalonData**[**i **-** 1**]));**

**}**

endStop **=** Math**.**Min**(**endStop**,** record**.**EthalonData**.**Count**);**

**if** **(**record**.**EthalonData**.**Count **<** 1**)**

endStop **=** timeList**.**Count**;**

double xCoord **=** 0**;**

// добавляем фейковые начальные точки

System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint ptFake1 **=** **new** System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint**();**

ptFake1**.**XValue **=** xCoord**;**

ptFake1**.**SetValueY**(**0**);**

xCoord **+=** xStep**;**

interruptSerie**.**Points**.**Add**(**ptFake1**);**

// теперь считаем все остальные точки

**for** **(**int i **=** 1**;** i **<** endStop**;** i**++)**

**{**

int pulseTime **=** timeList**[**i**]** **-** timeList**[**i **-** 1**];**

pulseTime **\*=** 100**;**

int pulseTimePercents **=** pulseTime **/** maxPulseTime**;**

pulseTimePercents **=** 100 **-** pulseTimePercents**;**

System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint pt **=** **new** System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint**();**

pt**.**XValue **=** xCoord**;**

pt**.**SetValueY**(**pulseTimePercents**);**

xCoord **+=** xStep**;**

interruptSerie**.**Points**.**Add**(**pt**);**

**}** // for

xCoord **=** 0**;**

// добавляем фейковые начальные точки

System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint ptFake2 **=** **new** System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint**();**

ptFake2**.**XValue **=** xCoord**;**

ptFake2**.**SetValueY**(**0**);**

xCoord **+=** xStep**;**

ethalonSerie**.**Points**.**Add**(**ptFake2**);**

// считаем график эталона

**for** **(**int i **=** 1**;** i **<** record**.**EthalonData**.**Count**;** i**++)**

**{**

int pulseTime **=** record**.**EthalonData**[**i**]** **-** record**.**EthalonData**[**i **-** 1**];**

pulseTime **\*=** 100**;**

int pulseTimePercents **=** pulseTime **/** maxPulseTime**;**

pulseTimePercents **=** 100 **-** pulseTimePercents**;**

System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint pt **=** **new** System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**DataPoint**();**

pt**.**XValue **=** xCoord**;**

pt**.**SetValueY**(**pulseTimePercents**);**

xCoord **+=** xStep**;**

ethalonSerie**.**Points**.**Add**(**pt**);**

**}** // for

**if** **(**modal**)**

**{**

vcf**.**ShowDialog**();**

**}**

**else**

**{**

vcf**.**Show**();**

vcf**.**BringToFront**();**

**}**

**}**

**private** void logDataGrid\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

var senderGrid **=** **(**DataGridView**)**sender**;**

**if** **(**senderGrid**.**Columns**[**e**.**ColumnIndex**]** **is** DataGridViewButtonColumn **&&**

e**.**RowIndex **>=** 0**)**

**{**

TreeNode child **=** treeView**.**SelectedNode**;**

string stationID **=** ""**;**

string stationName **=** ""**;**

**if(**child **!=** **null** **&&** child**.**Tag **is** ArchiveTreeLogItemRecord**)**

**{**

ArchiveTreeLogItemRecord ali **=** child**.**Tag **as** ArchiveTreeLogItemRecord**;**

ArchiveTreeRootItem atri **=** ali**.**Parent**.**Parent**;**

stationID **=** atri**.**GUID**;**

stationName **=** atri**.**GUID**;**

**if** **(**ControllerNames**.**Instance**.**Names**.**ContainsKey**(**stationID**))**

stationName **=** ControllerNames**.**Instance**.**Names**[**stationID**];**

**}**

ShowChart**(**senderGrid**.**Rows**[**e**.**RowIndex**].**Tag **as** InterruptRecord**,** stationID**,** stationName**,true);**

**}**

**}**

/\*

private void SaveEthalonToDisk(string dirName, List<int> data, string filename)

{

string resultfname = dirName + filename;

System.IO.BinaryWriter bw;

try

{

bw = new System.IO.BinaryWriter(new System.IO.FileStream(resultfname, System.IO.FileMode.Create));

for (int i = 0; i < data.Count; i++)

bw.Write(data[i]);

bw.Close();

}

catch

{

return;

}

}

\*/

**private** void btnSaveEthalons\_Click**(object** sender**,** EventArgs e**)**

**{**

/\*

if(folderBrowserDialog.ShowDialog() != DialogResult.OK)

{

return;

}

string dir = folderBrowserDialog.SelectedPath;

if (!dir.EndsWith("\\"))

dir += "\\";

dir += Config.Instance.ControllerGUID + "\\";

try

{

System.IO.Directory.CreateDirectory(dir);

}

catch

{

MessageBox.Show("Не получается создать папку \"" + dir + "\"!");

return;

}

SaveEthalonToDisk(dir, this.ethalon0UpData, "Канал 1, вверх.ETL");

SaveEthalonToDisk(dir, this.ethalon0DwnData, "Канал 1, вниз.ETL");

SaveEthalonToDisk(dir, this.ethalon1UpData, "Канал 2, вверх.ETL");

SaveEthalonToDisk(dir, this.ethalon1DwnData, "Канал 2, вниз.ETL");

SaveEthalonToDisk(dir, this.ethalon2UpData, "Канал 3, вверх.ETL");

SaveEthalonToDisk(dir, this.ethalon2DwnData, "Канал 3, вниз.ETL");

MessageBox.Show("Эталоны сохранены по адресу \"" + dir + "\".");

\*/

**}**

byte**[]** dataToSend **=** **null;**

bool inUploadFileToController **=** **false;**

**private** void SendEthalonData**()**

**{**

**this.**currentTransport**.**Write**(**dataToSend**,** dataToSend**.**Length**);**

inUploadFileToController **=** **false;**

**}**

**private** void FillEthalonWithData**(**List**<**int**>** etl**,** byte**[]** data**)**

**{**

etl**.**Clear**();**

byte**[]** dt **=** **new** byte**[**4**];**

**for** **(**int i **=** 0**;** i **<** data**.**Length**;** i **+=** 4**)**

**{**

**try**

**{**

dt**[**0**]** **=** data**[**i**];**

dt**[**1**]** **=** data**[**i **+** 1**];**

dt**[**2**]** **=** data**[**i **+** 2**];**

dt**[**3**]** **=** data**[**i **+** 3**];**

int curVal **=** BitConverter**.**ToInt32**(**dt**,** 0**);**

etl**.**Add**(**curVal**);**

**}**

**catch**

**{**

**break;**

**}**

**}**

**}**

**private** int uploadedEthalonChannel **=** 0**;**

**private** int uploadedEthalonRod **=** 0**;**

**private** void btnUploadEthalon\_Click**(object** sender**,** EventArgs e**)**

**{**

UploadFileDialog ufd **=** **new** UploadFileDialog**(this);**

**if(**ufd**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

string sourcefilename **=** ufd**.**GetSelectedFileName**();**

string targetfilename **=** ufd**.**GetTargetFileName**();**

**if** **(**sourcefilename**.**Length **<** 1**)**

**return;**

**try**

**{**

dataToSend **=** System**.**IO**.**File**.**ReadAllBytes**(**sourcefilename**);**

ShowWaitCursor**(true);**

uploadedEthalonChannel **=** Convert**.**ToInt32**(**ufd**.**nudChannelNumber**.**Value**)** **-** 1**;**

uploadedEthalonRod **=** ufd**.**cbRodMove**.**SelectedIndex**;**

inUploadFileToController **=** **true;**

PushCommandToQueue**(**SET\_PREFIX **+** "UPL|" **+** dataToSend**.**Length**.**ToString**()** **+** "|" **+** targetfilename**,** UploadEthalonCompleted**,** **null,** SendEthalonData**);**

**}**

**catch**

**{**

ShowWaitCursor**(false);**

**}**

**}**

**}**

**private** void UploadEthalonCompleted**(**Answer a**)**

**{**

ShowWaitCursor**(false);**

**if(**a**.**IsOkAnswer**)**

**{**

**switch(this.**uploadedEthalonChannel**)**

**{**

**case** 0**:**

**{**

**if** **(this.**uploadedEthalonRod **==** 0**)**

FillEthalonWithData**(this.**ethalon0UpData**,** dataToSend**);**

**else**

FillEthalonWithData**(this.**ethalon0DwnData**,** dataToSend**);**

**}**

**break;**

**case** 1**:**

**{**

**if** **(this.**uploadedEthalonRod **==** 0**)**

FillEthalonWithData**(this.**ethalon1UpData**,** dataToSend**);**

**else**

FillEthalonWithData**(this.**ethalon1DwnData**,** dataToSend**);**

**}**

**break;**

**case** 2**:**

**{**

**if** **(this.**uploadedEthalonRod **==** 0**)**

FillEthalonWithData**(this.**ethalon2UpData**,** dataToSend**);**

**else**

FillEthalonWithData**(this.**ethalon2DwnData**,** dataToSend**);**

**}**

**break;**

**}**

dataToSend **=** **null;**

MessageBox**.**Show**(**"Эталон загружен в контроллер!"**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

dataToSend **=** **null;**

MessageBox**.**Show**(**"Не удалось загрузить эталон в контроллер!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void MainForm\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

ControllerNames**.**Instance**.**Save**();**

**}**

**private** void btnControllerName\_Click**(object** sender**,** EventArgs e**)**

**{**

ControllerNameForm cnf **=** **new** ControllerNameForm**();**

cnf**.**ShowDialog**();**

TreeNode archiveNode **=** treeView**.**Nodes**[**1**];**

// переименовываем в архиве

**for** **(**int i **=** 0**;** i **<** archiveNode**.**Nodes**.**Count**;** i**++)**

**{**

TreeNode child **=** archiveNode**.**Nodes**[**i**];**

**if** **(**child**.**Tag **is** ArchiveTreeRootItem**)**

**{**

ArchiveTreeRootItem atri **=** child**.**Tag **as** ArchiveTreeRootItem**;**

**if** **(**atri**.**GUID **==** Config**.**Instance**.**ControllerGUID**)**

**{**

TreeNode existingNode **=** child**;**

**if** **(**ControllerNames**.**Instance**.**Names**.**ContainsKey**(**Config**.**Instance**.**ControllerGUID**))**

existingNode**.**Text **=** ControllerNames**.**Instance**.**Names**[**Config**.**Instance**.**ControllerGUID**];**

**else**

existingNode**.**Text **=** Config**.**Instance**.**ControllerGUID**;**

**break;**

**}**

**}**

**}** // for

setConnectionStatusMessage**();**

**}**

**private** void btnImportSettings\_Click**(object** sender**,** EventArgs e**)**

**{**

ArchiveImportForm af **=** **new** ArchiveImportForm**(this);**

af**.**ShowDialog**();**

**}**

**private** string archiveEthalonChartExportFileName**;**

**private** void ShowArchiveEthalon**(**ArchiveTreeEthalonItemRecord atei**)**

**{**

string fname **=** atei**.**FileName**;**

DateTime modification **=** System**.**IO**.**File**.**GetLastWriteTime**(**fname**);**

archiveEthalonChartExportFileName **=** modification**.**ToString**(**"yyyy-MM-dd HH.mm"**);**

**try**

**{**

List**<**byte**>** content **=** **new** List**<**byte**>(**System**.**IO**.**File**.**ReadAllBytes**(**fname**));**

CreateChart**(**content**,** **this.**archiveAthalonChart**);**

**this.**plArchiveEthalonChart**.**BringToFront**();**

**}**

**catch** **{** **}**

**}**

**private** async void DoReadArchiveLog**(**ConnectForm frm**,** string fname**)**

**{**

byte**[]** result**;**

**using** **(**System**.**IO**.**FileStream SourceStream **=** System**.**IO**.**File**.**Open**(**fname**,** System**.**IO**.**FileMode**.**Open**))**

**{**

result **=** **new** byte**[**SourceStream**.**Length**];**

await SourceStream**.**ReadAsync**(**result**,** 0**,** **(**int**)**SourceStream**.**Length**);**

**}**

//List<byte> content = new List<byte>(System.IO.File.ReadAllBytes(fname));

List**<**byte**>** content **=** **new** List**<**byte**>(**result**);**

ShowLogFile**(**content**,** **this.**archiveLogDataGrid**,** "1"**,** **false,**frm**,null,false);**

**this.**archiveLogDataGrid**.**BringToFront**();**

frm**.**DialogResult **=** DialogResult**.**OK**;**

**}**

**private** ArchiveTreeLogItemRecord archiveWorkRecord **=** **null;**

**private** void DoShowArchiveLog**(**ConnectForm frm**)**

**{**

System**.**Diagnostics**.**Debug**.**Assert**(**archiveWorkRecord **!=** **null);**

string fname **=** archiveWorkRecord**.**FileName**;**

frm**.**Update**();**

**try**

**{**

/\*

List<byte> content = new List<byte>(System.IO.File.ReadAllBytes(fname));

ShowLogFile(content, this.archiveLogDataGrid, "1", false);

this.archiveLogDataGrid.BringToFront();

\*/

DoReadArchiveLog**(**frm**,** fname**);**

**}**

**catch** **{**

frm**.**DialogResult **=** DialogResult**.**OK**;**

**}**

**}**

**private** void ShowArchiveLog**(**ArchiveTreeLogItemRecord atlir**)**

**{**

ConnectForm cn **=** **new** ConnectForm**(true);**

cn**.**OnConnectFormShown **=** **this.**DoShowArchiveLog**;**

cn**.**lblCurrentAction**.**Text **=** "Загружаем лог..."**;**

archiveWorkRecord **=** atlir**;**

cn**.**ShowDialog**();**

**}**

**private** void treeView\_MouseDoubleClick**(object** sender**,** MouseEventArgs e**)**

**{**

TreeNode selNode **=** treeView**.**SelectedNode**;**

**if** **(**selNode **==** **null)**

**return;**

**if(**selNode**.**Tag **is** ArchiveTreeEthalonItemRecord**)**

**{**

ShowArchiveEthalon**(**selNode**.**Tag **as** ArchiveTreeEthalonItemRecord**);**

**return;**

**}**

**if** **(**selNode**.**Tag **is** ArchiveTreeLogItemRecord**)**

**{**

ShowArchiveLog**(**selNode**.**Tag **as** ArchiveTreeLogItemRecord**);**

**return;**

**}**

**}**

**private** bool inSetBordersToController **=** **true;**

**private** void btnSetBorders\_Click**(object** sender**,** EventArgs e**)**

**{**

inSetBordersToController **=** **true;**

ShowWaitCursor**(true);**

string s **=** ""**;**

s **+=** Convert**.**ToString**(**nudLowBorder**.**Value**)** **+** PARAM\_DELIMITER**;**

s **+=** Convert**.**ToString**(**nudHighBorder**.**Value**);**

PushCommandToQueue**(**SET\_PREFIX **+** "TBORDERS" **+** PARAM\_DELIMITER **+** s**,** ParseSetBorders**);**

**}**

**private** bool inSetRelayDelayToController **=** **true;**

**private** void btnSetRelayDelay\_Click**(object** sender**,** EventArgs e**)**

**{**

inSetRelayDelayToController **=** **true;**

ShowWaitCursor**(true);**

string s **=** ""**;**

s **+=** Convert**.**ToString**(**nudRelayDelay**.**Value**);**

s **+=** PARAM\_DELIMITER **+** Convert**.**ToString**(**nudACSDelay**.**Value**);**

PushCommandToQueue**(**SET\_PREFIX **+** "RDELAY" **+** PARAM\_DELIMITER **+** s**,** ParseSetRelayDelay**);**

**}**

**private** void archiveLogDataGrid\_CellValueNeeded**(object** sender**,** DataGridViewCellValueEventArgs e**)**

**{**

DataGridView targetGrid **=** sender **as** DataGridView**;**

int rowNumber **=** e**.**RowIndex**;**

**if** **(**rowNumber **>=** targetGrid**.**RowCount**)**

**return;**

**if** **(!**gridToListCollection**.**ContainsKey**(**targetGrid**))**

**return;**

LogInfo linf **=** gridToListCollection**[**targetGrid**];**

**if** **(**rowNumber **>=** linf**.**list**.**Count**)**

**return;**

InterruptRecord record **=** linf**.**list**[**rowNumber**];**

DataGridViewRow row **=** targetGrid**.**Rows**[**rowNumber**];**

row**.**Tag **=** record**;**

row**.**DefaultCellStyle**.**BackColor **=** rowNumber **%** 2 **==** 0 **?** Color**.**LightGray **:** Color**.**White**;**

**if(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Num"**))**

**{**

e**.**Value **=** **(**rowNumber **+** 1**).**ToString**();**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Time"**))**

**{**

e**.**Value **=** record**.**InterruptInfo**.**InterruptTime**.**ToString**(**"dd.MM.yyyy HH:mm:ss"**);**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Temp"**))**

**{**

e**.**Value **=** record**.**InterruptInfo**.**SystemTemperature**.**ToString**(**"0.00"**)** **+** " °C"**;**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Channel"**))**

**{**

e**.**Value **=** **(**1 **+** record**.**ChannelNumber**).**ToString**();**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Rod"**))**

**{**

e**.**Value **=** EnumHelpers**.**GetEnumDescription**(**record**.**RodPosition**);**

**if** **(**record**.**RodPosition **==** RodPosition**.**Broken**)**

row**.**Cells**[**e**.**ColumnIndex**].**Style**.**BackColor **=** Color**.**LightSalmon**;**

**else**

row**.**Cells**[**e**.**ColumnIndex**].**Style**.**BackColor **=** Color**.**White**;**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Compare"**))**

**{**

e**.**Value **=** EnumHelpers**.**GetEnumDescription**(**record**.**EthalonCompareResult**);**

**if** **(**record**.**EthalonCompareResult **==** EthalonCompareResult**.**MatchEthalon**)**

row**.**Cells**[**e**.**ColumnIndex**].**Style**.**BackColor **=** Color**.**LightGreen**;**

**else**

row**.**Cells**[**e**.**ColumnIndex**].**Style**.**BackColor **=** Color**.**LightSalmon**;**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Etl"**))**

**{**

e**.**Value **=** EnumHelpers**.**GetEnumDescription**(**record**.**EthalonCompareNumber**);**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Motoresource"**))**

**{**

**if(**linf**.**computeMotoresurcePercents**)**

**{**

int resMax **=** 1**;**

**switch** **(**record**.**ChannelNumber**)**

**{**

//DEPRECATED: case 0:

**default:**

resMax **=** Config**.**Instance**.**MotoresourceMax1**;**

**break;**

**}**

**if** **(**resMax **<** 1**)**

resMax **=** 1**;**

float motoPercents **=** **(**record**.**Motoresource **\*** 100.0f**)** **/** resMax**;**

e**.**Value **=** record**.**Motoresource**.**ToString**()** **+** " (" **+** motoPercents**.**ToString**(**"0.00"**)** **+** "%)"**;**

**}**

**else**

**{**

e**.**Value **=** record**.**Motoresource**.**ToString**();** **;**

**}**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Pulses"**))**

**{**

e**.**Value **=** record**.**InterruptData**.**Count**.**ToString**();**

**}**

**else** **if** **(**targetGrid**.**Columns**[**e**.**ColumnIndex**].**Name**.**StartsWith**(**"Btn"**))**

**{**

e**.**Value **=** "Просмотр"**;**

**}**

**}**

**private** void btnExportEthalonToImage\_Click**(object** sender**,** EventArgs e**)**

**{**

saveFileDialog**.**FileName **=** archiveEthalonChartExportFileName**;**

**if** **(**saveFileDialog**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

bool errors **=** **false;**

string fname **=** saveFileDialog**.**FileName**;**

**switch** **(**saveFileDialog**.**FilterIndex**)**

**{**

**case** 1**:** // png

**{**

**try**

**{**

**this.**archiveAthalonChart**.**SaveImage**(**fname**,** System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**ChartImageFormat**.**Png**);**

**}**

**catch**

**{**

errors **=** **true;**

**}**

**}**

**break;**

**case** 2**:** // jpeg

**{**

**try**

**{**

**this.**archiveAthalonChart**.**SaveImage**(**fname**,** System**.**Windows**.**Forms**.**DataVisualization**.**Charting**.**ChartImageFormat**.**Jpeg**);**

**}**

**catch**

**{**

errors **=** **true;**

**}**

**}**

**break;**

**}** // switch

**if** **(**errors**)**

**{**

MessageBox**.**Show**(**"Ошибка экспорта!"**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**else**

**{**

MessageBox**.**Show**(**"Картинка экспортирована."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**}**

**}**

**private** void treeView\_AfterSelect**(object** sender**,** TreeViewEventArgs e**)**

**{**

TreeNode selectedNode **=** e**.**Node**;**

**if** **(**selectedNode **==** **null** **||** selectedNode**.**Tag **==** **null)**

**{**

ShowStartPanel**();**

**return;**

**}**

**if** **(**selectedNode**.**Tag **is** TreeNodeType**)**

**{**

TreeNodeType tp **=** **(**TreeNodeType**)**selectedNode**.**Tag**;**

**switch** **(**tp**)**

**{**

**case** TreeNodeType**.**MainSettingsNode**:**

ShowMainSettings**();**

**break;**

//TODO: Тут другие панели!!!

**case** TreeNodeType**.**SDSettingsNode**:**

**this.**plSDSettings**.**BringToFront**();**

**break;**

**}** // switch

**}**

**}**

**private** bool inSetEthalonRecordToController **=** **false;**

**private** void btnRecordEthalonUp\_Click**(object** sender**,** EventArgs e**)**

**{**

recordEthalon**(**"UP"**);**

**}**

**private** void btnRecordEthalonDown\_Click**(object** sender**,** EventArgs e**)**

**{**

recordEthalon**(**"DOWN"**);**

**}**

**private** void recordEthalon**(**string dir**)**

**{**

ShowWaitCursor**(true);**

inSetEthalonRecordToController **=** **true;**

PushCommandToQueue**(**GET\_PREFIX **+** "EREC" **+** PARAM\_DELIMITER **+** dir**,** ParseRecordEthalon**);**

**}**

**private** void ParseRecordEthalon**(**Answer a**)**

**{**

inSetEthalonRecordToController **=** **false;**

ShowWaitCursor**(false);**

**if** **(**a**.**IsOkAnswer**)**

**{**

MessageBox**.**Show**(**"Эталон успешно записан."**,** "Сообщение"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

**else**

**{**

MessageBox**.**Show**(**"Ошибка записи эталона!"**,** "Ошибка"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void btnDisconnect\_Click**(object** sender**,** EventArgs e**)**

**{**

currentTransport**.**Disconnect**();**

**this.**treeView**.**Nodes**[**0**].**Nodes**.**Clear**();**

**}**

**private** void btnConnect\_Click**(object** sender**,** EventArgs e**)**

**{**

DoConnect**(**""**,** **false,** **true);**

**}**

**}**

**}**