```
///-----------------------------------------------------------------------------
///
///  File Name:         frmAbout.cs
///
///  File Description: This file implements all of the functionality of the
///                    ISE Manipulator's about form.  This file contains
///                    only the code for the about form and nothing else.
///                    This code has been developed to assist Team ISE in
///                    working with JPEG images and testing techniques used
///                    to develop our Selective Encryption algorithm for ISO
///                    Standard Baseline JPEG Image files.
///
///  Project Name:     Selective Encryption for JPEG Images
///                    CSCI 4308-4318: Senior Project
///                    August 2003 to May 2004
///                    Department of Computer Science
///                    University of Colorado at Boulder
///
///  Project Sponsor:  Tom Lookabaugh
///                    Assistant Professor of Computer Science
///                    University of Colorado at Boulder
///
///  Project Manager:  Bruce Sanders
///                    University of Colorado at Boulder
///
///  Team ISE Members: Shinya Daigaku
///                    Geoffrey Griffith
///                    Joe Jarchow
///                    Joseph Kadhim
///                    Andrew Pouzeshi
///
///-----------------------------------------------------------------------------
///
///  This code is open source and may be used with no cost.
///  The authors are in no way responsible for any effects
///  from the usage of this code.  It is provided as is with
///  no warranties, protections, promises or any form of
///  support.  The authors would hope it would only be used
///  for good purposes.  Thank you.
///
///-----------------------------------------------------------------------------
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;


namespace JPEG_Manipulator
{
  /// <summary>
  /// Summary description for frmAbout.
  /// </summary>
  public class frmAbout : System.Windows.Forms.Form
  {
    private System.Windows.Forms.PictureBox picAbout;
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;


    /// <summary>
    /// This is the frmAbout() constructor.
    /// </summary>
    public frmAbout()
    {
      InitializeComponent();
    }
```



```

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
        if(components != null)
        {
          components.Dispose();
        }
      }
      base.Dispose( disposing );
    }

    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      System.Resources.ResourceManager resources =
                  new System.Resources.ResourceManager(typeof(frmAbout));
      this.picAbout = new System.Windows.Forms.PictureBox();
      this.SuspendLayout();
      //
      // picAbout
      //
      this.picAbout.Image =
                  ((System.Drawing.Image)(resources.GetObject("picAbout.Image"))
);
      this.picAbout.Location = new System.Drawing.Point(8, 8);
      this.picAbout.Name = "picAbout";
      this.picAbout.Size = new System.Drawing.Size(448, 608);
      this.picAbout.SizeMode =
                  System.Windows.Forms.PictureBoxSizeMode.StretchImage;
      this.picAbout.TabIndex = 0;
      this.picAbout.TabStop = false;
      this.picAbout.Click += new System.EventHandler(this.picAbout_Click);
      //
      // frmAbout
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
      this.ClientSize = new System.Drawing.Size(464, 621);
      this.Controls.Add(this.picAbout);
      this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
      this.Name = "frmAbout";
      this.StartPosition =
                  System.Windows.Forms.FormStartPosition.CenterScreen;
      this.Text = "About the ISE JPEG Manipulator";
      this.TopMost = true;
      this.ResumeLayout(false);

    }
    #endregion

    private void picAbout_Click(object sender, System.EventArgs e)
    {
      this.Close();
    }
  }
}
```

```
///-----------------------------------------------------------------------
///
///  File Name:         frmLoad.cs
///
///  File Description: This file implements all of the functionality of the
///                    ISE Manipulator's loading form.  This file contains
///                    only the code for the loading form and nothing else.
///                    This code has been developed to assist Team ISE in
///                    working with JPEG images and testing techniques used
///                    to develop our Selective Encryption algorithm for ISO
///                    Standard Baseline JPEG Image files.
///
///  Project Name:     Selective Encryption for JPEG Images
///                    CSCI 4308-4318: Senior Project
///                    August 2003 to May 2004
///                    Department of Computer Science
///                    University of Colorado at Boulder
///
///  Project Sponsor:  Tom Lookabaugh
///                    Assistant Professor of Computer Science
///                    University of Colorado at Boulder
///
///  Project Manager:  Bruce Sanders
///                    University of Colorado at Boulder
///
///  Team ISE Members: Shinya Daigaku
///                    Geoffrey Griffith
///                    Joe Jarchow
///                    Joseph Kadhim
///                    Andrew Pouzeshi
///
///-----------------------------------------------------------------------
///
///  This code is open source and may be used with no cost.
///  The authors are in no way responsible for any effects
///  from the usage of this code.  It is provided as is with
///  no warranties, protections, promises or any form of
///  support.  The authors would hope it would only be used
///  for good purposes.  Thank you.
///
///-----------------------------------------------------------------------
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;


namespace JPEG_Manipulator
{
  /// <summary>
  /// Summary description for frmLoadMessage.
  /// </summary>
  public class frmLoad : System.Windows.Forms.Form
  {
    private System.Windows.Forms.ProgressBar barLoadProgress;
    private System.Windows.Forms.Label lblLoad;
    private System.Windows.Forms.Button btnCancelLoad;

    private bool canceled;


    #region Form Required Code

    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;
```



```
    public frmLoad()
    {
      InitializeComponent();
      LoadFormConstructor();
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
        if(components != null)
        {
          components.Dispose();
        }
      }
      base.Dispose( disposing );
    }

    #endregion


    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      System.Resources.ResourceManager resources =
                new System.Resources.ResourceManager(typeof(frmLoad));
      this.barLoadProgress = new System.Windows.Forms.ProgressBar();
      this.lblLoad = new System.Windows.Forms.Label();
      this.btnCancelLoad = new System.Windows.Forms.Button();
      this.SuspendLayout();
      //
      // barLoadProgress
      //
      this.barLoadProgress.Location = new System.Drawing.Point(8, 32);
      this.barLoadProgress.Name = "barLoadProgress";
      this.barLoadProgress.Size = new System.Drawing.Size(272, 23);
      this.barLoadProgress.TabIndex = 0;
      //
      // lblLoad
      //
      this.lblLoad.Location = new System.Drawing.Point(16, 8);
      this.lblLoad.Name = "lblLoad";
      this.lblLoad.Size = new System.Drawing.Size(256, 16);
      this.lblLoad.TabIndex = 1;
      this.lblLoad.Text = "Data Loading, Please Wait...";
      //
      // btnCancelLoad
      //
      this.btnCancelLoad.Cursor = System.Windows.Forms.Cursors.Arrow;
      this.btnCancelLoad.Location = new System.Drawing.Point(88, 64);
      this.btnCancelLoad.Name = "btnCancelLoad";
      this.btnCancelLoad.Size = new System.Drawing.Size(112, 24);
      this.btnCancelLoad.TabIndex = 0;
      this.btnCancelLoad.Text = "&Cancel Load";
      this.btnCancelLoad.Click += new
            System.EventHandler(this.btnCancelLoad_Click);
      //
      // frmLoad
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
      this.ClientSize = new System.Drawing.Size(292, 93);
      this.Controls.Add(this.btnCancelLoad);
```

```
      this.Controls.Add(this.lblLoad);
      this.Controls.Add(this.barLoadProgress);
      this.Cursor = System.Windows.Forms.Cursors.WaitCursor;
      this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
      this.Name = "frmLoad";
      this.StartPosition =
              System.Windows.Forms.FormStartPosition.CenterScreen;
      this.Text = "Loading Data";
      this.TopMost = true;
      this.ResumeLayout(false);

    }
    #endregion


    /// <summary>
    /// If this button is clicked, the Cancelled property on this form
    /// will be set to true.  This property will remain true until the
    /// is destroyed.
    /// </summary>
    /// <param name="sender">The sender parameter is a pointer to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnCancelLoad_Click(object sender, System.EventArgs e)
    {
      canceled = true;
    }


    /// <summary>
    /// This is the constructor that ISE will initialize all our variables
    /// for this form and then this method will be called by this Load form
    /// constructor, in this file.
    /// </summary>
    private void LoadFormConstructor()
    {
      canceled = false;
      StartLoading(0, 100, 1);
      this.barLoadProgress.Value = 0;
      this.ShowInTaskbar = true;
    }


    /// <summary>
    /// True if the Cancel Button has been hit.
    /// </summary>
    public bool Canceled
    {
      get { return canceled; }
      set { canceled = value; }
    }


    /// <summary>
    /// Gets or Sets the value of the Progress Bar.
    /// </summary>
    public int LoadProgressValue
    {
      get { return barLoadProgress.Value; }
      set { barLoadProgress.Value = value; }
    }


    /// <summary>
    /// This resets and prepares the Load form.
    /// </summary>
    /// <param name="MinValue">Minimum value for the Load Bar.</param>
    /// <param name="MaxValue">Maximum value for the Load Bar.</param>
```



```
    /// <param name="StepSize">Step size for the Load Bar.</param>
    public void StartLoading(int MinValue, int MaxValue, int StepSize)
    {
      int i = 0;
      this.barLoadProgress.Maximum = MaxValue;
      this.barLoadProgress.Minimum = MinValue;
      this.barLoadProgress.Step = StepSize;

      if(i < MinValue) i = MinValue;
      this.barLoadProgress.Value = i;
      this.barLoadProgress.Update();
      this.Show();
      this.Activate();
      this.btnCancelLoad.Focus();
    }


    /// <summary>
    /// This function updates the progress bar.  If the been cancel button
    /// has been clicked, then this function will return false, but form will
    /// STILL be updated.
    /// </summary>
    /// <returns>Returns true if cancel button has NOT been pressed.</returns>
    public bool UpdateForm()
    {
      this.Update();
      if(canceled)
      {
        if(MessageBox.Show(
          "Are you sure you want to CANCEL this operation?\n" +
          "Clicking \"OK\" will cancel this operation.\n" +
          "Clicking \"CANCEL\" will continue this operation.\n",
          "Operation Aborted!",
          MessageBoxButtons.OKCancel,
          MessageBoxIcon.Error) == DialogResult.OK)
        {
          canceled = true;
        }
        else canceled = false;

      }
      return !canceled;
    }


    /// <summary>
    /// This function updates and increments the progress bar.  If the been
    /// cancel button has been clicked, then this function will return false,
    /// but form will STILL be updated and incremented.
    /// </summary>
    /// <returns>Returns true if cancel button has NOT been pressed.</returns>
    public bool UpdateAndIncrement()
    {
      this.barLoadProgress.PerformStep();
      this.Update();
      return !canceled;
    }


  }
}
```

```
///-----------------------------------------------------------------------
///
///  File Name:        frmMain.cs
///
///  File Description: This file implements all of the functionality of the
///                    ISE Manipulator's main form.  This file contains the
///                    all of the code for the ISE Manipulator, except the
///                    code for the "About Form" (frmAbout.cs) and the code
///                    for the "Loading Form" (frmLoading.cs).  This code
///                    has been developed to assist Team ISE in working with
///                    JPEG images and testing techniques used to develop
///                    our Selective Encryption algorithm for ISO Standard
///                    Baseline JPEG Image files.
///
///  Project Name:     Selective Encryption for JPEG Images
///                    CSCI 4308-4318: Senior Project
///                    August 2003 to May 2004
///                    Department of Computer Science
///                    University of Colorado at Boulder
///
///  Project Sponsor:  Tom Lookabaugh
///                    Assistant Professor of Computer Science
///                    University of Colorado at Boulder
///
///  Project Manager:  Bruce Sanders
///                    Assistant Professor of Computer Science
///                    University of Colorado at Boulder
///
///  Team ISE Members: Shinya Daigaku
///                    Geoffrey Griffith
///                    Joe Jarchow
///                    Joseph Kadhim
///                    Andrew Pouzeshi
///
///-----------------------------------------------------------------------
///
///  This code is open source and may be used with no cost.
///  The authors are in no way responsible for any effects
///  from the usage of this code.  It is provided as is with
///  no warranties, protections, promises or any form of
///  support.  The authors would hope it would only be used
///  for good purposes.  Thank you.
///
///-----------------------------------------------------------------------
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Text;


namespace JPEG_Manipulator
{

  /// <summary>
  /// This is the JPEG Manipulator's main form inhereited from the
  /// System.Windows.Forms.Form class.  This form provides most of the
  /// functionality required for breaking down JPEG images, loading them
  /// into the interface, allowing the user to alter values, and recreate
  /// a new image based upon the current value loaded.
  /// </summary>
  public class frmMain : System.Windows.Forms.Form
  {
    public const string VERSION = "1.0.7";
```



```
    #region ISE Coded Functions

    #region ISE JPEG Manipulator Variables and Constructor

    // Data member for the about form
    private System.Windows.Forms.Form MainAbout;
    private frmLoad Loading;
    private frmSplash SplashScreen;

    // Data memebers for Loaded JPEG images
    private System.Drawing.Image JPEG;
    private System.Drawing.Image ISE;
    private System.Drawing.Image JPEGsmall;
    private System.Drawing.Image ISEsmall;

    // Data member to store the JPEG image file order
    private Queue FileOrder;

    // Data members for the raw JPEG image data
    //
    // The Max file size is hard coded for now
    private const int MAX_BYTES = 10485760;    // 10 meg
    private const int MAX_FILE_SIZE = 20971520;  // 20 meg (2x MAX_BYTES)
    private const int AVE_FILE_SIZE = 10485760;  // 10 meg

    // Assumes no more tables than the Baseline Compression
    private const int MAX_HUFFMAN = 8;
    private const int MAX_QUANTIZER = 4;
    private const int MAX_APPDATA = 10;

    // Data members for the original and new raw data stream
    private string OriginalEncodedData;
    private StringBuilder OriginalDataStream;
    private StringBuilder EncodedData;
    private byte[] NewData;

    // Fixed size variables
    private int NumberOfLines;
    private int RestartInterval;
    private int FileSize;
    private int ExpandImage;
    private int RestartMod8;

    private int SizeOfScanHeader;
    private int SizeOfProgression;
    private int SizeOfComments;

    private int[] SizeOfHuffman = new int[MAX_HUFFMAN];
    private int[] SizeOfQuantizer = new int[MAX_QUANTIZER];
    private int[] SizeOfAppData = new int[MAX_APPDATA];


    // Temporary Variables
    private int FrameSize;
    private int Count;
    private int Temp;
    private int Value;
    private int High;
    private int Low;
    private int temp;

    private string ProgramDirectory;


    // Others
    private FileStream OriginalFile;
    private FileStream NewFile;
    private string ManipulatedFileName;
```

```
  private bool LoadingInterface;

  // Random Number Generator
  private System.Random RandomNumber;

  // Data members to determine if the image is stretched
  private bool PicOriginalStretched;
  private bool PicOriginalSmallStretched;
  private bool PicManipulatedStretched;
  private System.Windows.Forms.Timer timerSplash;
  private System.Windows.Forms.MenuItem menuItem2;
  private System.Windows.Forms.MenuItem menuTutorial;
  private System.Windows.Forms.MenuItem menuManual;
  private System.Windows.Forms.MenuItem menuItem6;
  private System.Windows.Forms.MenuItem menuAbout;
  private bool PicManipulatedSmallStretched;


  /// <summary>
  /// Pre-conditions:    None.
  /// Post-conditions:
  ///    ISE variables and initialization routines have been executed.
  ///  Parameters:      None.
  ///  Return values:
  ///    Function returns void.
  ///  Description:
  ///    This function is used to execute all ISE initialization
  ///    logic.  This includes initialization routines for variables
  ///    and setting defaults.
  /// </summary>
  private void ISEConstructor()
  {
    if(FileOrder != null) FileOrder = null;
    FileOrder = new Queue();

    if(OriginalDataStream != null) OriginalDataStream = null;
    if(EncodedData != null) EncodedData = null;
    OriginalDataStream = new
      StringBuilder(AVE_FILE_SIZE, MAX_FILE_SIZE);
    EncodedData = new StringBuilder(AVE_FILE_SIZE, MAX_FILE_SIZE);

    LoadingInterface = false;

    NumberOfLines = 0;
    RestartInterval = 0;
    FileSize = 0;
    ExpandImage = 0;
    RestartMod8 = 0;

    RandomNumber = new
      System.Random(unchecked((int)DateTime.Now.Ticks));
    RandomNumber.Next(5000);

    PicOriginalStretched = false;
    PicOriginalSmallStretched = false;
    PicManipulatedStretched = false;
    PicManipulatedSmallStretched = false;

    ProgramDirectory = Environment.CurrentDirectory;

    // Update frmMain Text
    this.Text = "ISE JPEG Manipulator - Version " + VERSION;
  }


  #endregion ISE JPEG Manipulator Variables


  #region Interface Methods
```



```
  /// <summary>
  /// Pre-conditions:
  ///    The menuOpen menu object has generated a Click event.
  ///  Post-conditions:
  ///    A new original JPEG image has been loaded and displayed
  ///    within the picOriginal and the picOrignalSmall PictureBox
  ///    controls.
  ///  Description:
  ///    This method is used to resolve a Click event generated by
  ///    the menuOpen menu object.  The purpose of this menu object
  ///    is to allow the user to open a new original JPEG image file
  ///    within the application.  This function will simply call the
  ///    LoadNewPicture() function described in section 4.2.3.2 of
  ///    this document.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass
  /// event data.</param>
  private void menuOpen_Click(object sender, System.EventArgs e)
  {
    LoadNewPicture();
  }  // End of: menuOpen_Click(object sender, System.EventArgs e);


  /// <summary>
  /// Pre-conditions:
  ///    The menuExit menu object has generated a Click event.
  ///  Post-conditions:
  ///    The application is terminated and exited successfully.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    menuExit menu object.  The purpose of this menu object is to
  ///    allow the user to exit the application when they have
  ///    finished.  This function should check to see if there is any
  ///    unsaved data before exiting and if so, should ask the user
  ///    if they want to save the current information.  Then, this
  ///    function will call the Application.Exit() method to
  ///    successfully exit the Windows application.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass
  /// event data.</param>
  private void menuExit_Click(object sender, System.EventArgs e)
  {
    Application.Exit();
  }


  /// <summary>
  /// Pre-conditions:
  ///    The menuAbout menu object has generated a Click event.
  ///  Post-conditions:
  ///    The frmAbout Form has been displayed for the user to view.
  ///  Description:
  ///    This method is used to resolve a Click event generated by
  ///    the menuAbout menu object.  The purpose of this menu object
  ///    is to allow the user to view the about window to find out
  ///    details about the system.  This function creates a new
  ///    instance of the frmAbout form and then displays it for the
  ///    user.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass
  /// event data.</param>
```

```
  private void menuAbout_Click(object sender, System.EventArgs e)
  {
    MainAbout = new frmAbout();
    MainAbout.Show();
  }


  /// <summary>
  /// Pre-conditions:
  ///    The menuNewProject menu object has generated a Click event.
  ///  Post-conditions:
  ///    A new project file has been created by the application.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    menuNewProject menu object.  The purpose of this menu object
  ///    is to allow the user to create a new project file that will
  ///    allow them to store picture, note data and manipulated data
  ///    of original images. This function should check to see if
  ///    there is any unsaved data before creating a new project and
  ///    if so, should ask the user if they want to save the current
  ///    information.  This function should simply call the
  ///    CreateNewProject() method outlined in section 4.2.3.11 of
  ///    this document.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass
  /// event data.</param>
  private void menuNewProject_Click(object sender, System.EventArgs e)
  {
    ClearInterfaceData();
  }


  /// <summary>
  /// Pre-conditions:
  ///    The menuOpenProject menu object has generated a Click event.
  ///  Post-conditions:
  ///    A previously created project file has been opened by the
  ///    application and all values previously saved within the project
  ///    have been reloaded into the application interface.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    menuOpenProject menu object.  The purpose of this menu object is
  ///    to allow the user to open a previously created project file.
  ///    This function should check to see if there is any unsaved data
  ///    before creating a new project and if so, should ask the user if
  ///    they want to save the current information.  The values stored in
  ///    the project file will be reloaded into the application interface.
  ///    This function should simply call the LoadNewProject() method
  ///    outlined in section 4.2.3.9 of this document.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void menuOpenProject_Click(object sender, System.EventArgs e)
  {
    LoadNewProject();
  }


  /// <summary>
  /// Pre-conditions:
  ///    The menuSaveProject menu object has generated a Click event.
  ///  Post-conditions:
  ///    This function saves the current values loaded in the Manipulator,
  ///    project notes and any manipulate data values and stores them in
  ///    an SEP file.
```



```
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    menuOpenProject menu object.  The purpose of this menu object is
  ///    to allow the user to save the current project file, including the
  ///    original picture, manipulated picture and any notes included in
  ///    the project.  This function will simply call the SaveNewProject()
  ///    function described later in this document.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void menuSaveProject_Click(object sender, System.EventArgs e)
  {
    SaveNewProject();
  }


  /// <summary>
  /// Pre-conditions:
  ///    The txtManipulatedFile TextBox object has generated a TextChanged
  ///    event.
  ///  Post-conditions:
  ///    A warning is displayed if the changed text reflects a file path
  ///    that already exists.
  ///  Description:
  ///    This method is used to resolve a TextChanged event generated by
  ///    the txtManipulatedFile TextBox object.  The purpose of this
  ///    TextBox is to allow the user to specify the name and path of the
  ///    file that will be created, if the user chooses to create a
  ///    manipulated image.  This function checks to see if the file name
  ///    and path already exist, and if so, calls the ShowWarning()
  ///    function (described later in this document) to display a warning
  ///    to the users.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void txtManipulatedFile_TextChanged(object sender,
              System.EventArgs e)
  {
    bool Check;

    if(File.Exists(txtManipulatedFile.Text) &&
      (txtManipulatedFile.Text != txtOriginalFile.Text))
    {
      Check = ShowWarning(
        "File name: " + txtManipulatedFile.Text +
        "\nALREADY EXISTS!\nAre you sure you want to overwrite this file?",
        "File Exists");

      if(Check) ManipulatedFileName = txtManipulatedFile.Text;
      else txtManipulatedFile.Text = ManipulatedFileName;
    }
    else if(txtManipulatedFile.Text == txtOriginalFile.Text)
    {
      // Create a name for the changed file
      ManipulatedFileName = openFileDialog.FileName;
      string ttt = ManipulatedFileName.ToLower();
      ManipulatedFileName = ManipulatedFileName.ToLower();
      Count = ttt.IndexOf(".jpg");

      // Manipulated the file name if it already exists
      ManipulatedFileName = ManipulatedFileName.Insert(Count, "_changed0");
      Temp = 0;
      string num_length;
      while(File.Exists(ManipulatedFileName))
      {
```

```
        Count = ManipulatedFileName.IndexOf(Temp.ToString() + ".jpg");
        num_length = Temp.ToString();
        ManipulatedFileName =
                        ManipulatedFileName.Remove(Count, num_length.Length);
        Temp++;
        ManipulatedFileName =
                        ManipulatedFileName.Insert(Count, Temp.ToString());
      }

      txtManipulatedFile.Text = ManipulatedFileName;
      this.Update();
    }
    else ManipulatedFileName = txtManipulatedFile.Text;
  }


  /// <summary>
  /// Pre-conditions:
  ///    The txtQuantizer1 TextBox object has generated a Click event.
  ///  Post-conditions:
  ///    If this is the first time the data has been altered, the data is
  ///    copied into the txtQuantizerOriginal1 TextBox.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    txtQuantizer1 TextBox object.  The purpose of this TextBox is to
  ///    allow the user to manipulate the values in the first Quantizer
  ///    table contained within the JPEG image.  If this is the first time
  ///    this data has been altered, this function copies the data from
  ///    the txtQuantizer1 TextBox (before it has been changed) into the
  ///    txtQuantizerOriginal1 TextBox.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void txtQuantizer1_Click(object sender, System.EventArgs e)
  {
    if(!LoadingInterface && this.txtQuantizerOriginal1.Text == "")
    {
      txtQuantizerOriginal1.Text = txtQuantizer1.Text;
      lblQuantizerOriginalMarker1.Text = lblQuantizerMarker1.Text;
    }
  }


  /// <summary>
  /// Pre-conditions:
  ///    The txtQuantizer2 TextBox object has generated a Click event.
  ///  Post-conditions:
  ///    If this is the first time the data has been altered, the data is
  ///    copied into the txtQuantizerOriginal2 TextBox.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    txtQuantizer2 TextBox object.  The purpose of this TextBox is to
  ///    allow the user to manipulate the values in the second Quantizer
  ///    table contained within the JPEG image.  If this is the first time
  ///    this data has been altered, this function copies the data from the
  ///    txtQuantizer2 TextBox (before it has been changed) into the
  ///    txtQuantizerOriginal2 TextBox.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void txtQuantizer2_Click(object sender, System.EventArgs e)
  {
    if(!LoadingInterface && this.txtQuantizerOriginal2.Text == "")
    {
      txtQuantizerOriginal2.Text = txtQuantizer2.Text;
```



```
      lblQuantizerOriginalMarker2.Text = lblQuantizerMarker2.Text;
    }
  }


  /// <summary>
  /// Pre-conditions:
  ///    The txtQuantizer3 TextBox object has generated a Click event.
  ///  Post-conditions:
  ///    If this is the first time the data has been altered, the data is
  ///    copied into the txtQuantizerOriginal3 TextBox.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    txtQuantizer3 TextBox object.  The purpose of this TextBox is to
  ///    allow the user to manipulate the values in the third Quantizer
  ///    table contained within the JPEG image.  If this is the first time
  ///    this data has been altered, this function copies the data from the
  ///    txtQuantizer3 TextBox (before it has been changed) into the
  ///    txtQuantizerOriginal3 TextBox.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void txtQuantizer3_Click(object sender, System.EventArgs e)
  {
    if(!LoadingInterface && this.txtQuantizerOriginal3.Text == "")
    {
      txtQuantizerOriginal3.Text = txtQuantizer3.Text;
      lblQuantizerOriginalMarker3.Text = lblQuantizerMarker3.Text;
    }
  }


  /// <summary>
  /// Pre-conditions:
  ///    The txtQuantizer4 TextBox object has generated a Click event.
  ///  Post-conditions:
  ///    If this is the first time the data has been altered, the data is
  ///    copied into the txtQuantizerOriginal4 TextBox.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    txtQuantizer4 TextBox object.  The purpose of this TextBox is to
  ///    allow the user to manipulate the values in the fourth Quantizer
  ///    table contained within the JPEG image.  If this is the first time
  ///    this data has been altered, this function copies the data from the
  ///    txtQuantizer4 TextBox (before it has been changed) into the
  ///    txtQuantizerOriginal4 TextBox.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void txtQuantizer4_Click(object sender, System.EventArgs e)
  {
    if(!LoadingInterface && this.txtQuantizerOriginal4.Text == "")
    {
      txtQuantizerOriginal4.Text = txtQuantizer4.Text;
      lblQuantizerOriginalMarker4.Text = lblQuantizerMarker4.Text;
    }
  }


  /// <summary>
  /// Pre-conditions:
  ///    The txtHuffman1 TextBox object has generated a Click event.
  ///  Post-conditions:
  ///    If this is the first time the data has been altered, the data is
  ///    copied into the txtHuffmanOriginal1 TextBox.
```

```
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    txtHuffman1 TextBox object.  The purpose of this TextBox is to
    ///    allow the user to manipulate the values in the first Huffman table
    ///    contained within the JPEG image.  If this is the first time this
    ///    data has been altered, this function copies the data from the
    ///    txtHuffman1 TextBox (before it has been changed) into the
    ///    txtHuffmanOriginal1 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void txtHuffman1_GotFocus(object sender, System.EventArgs e)
    {
      if(!LoadingInterface && this.txtHuffmanOriginal1.Text == "")
      {
        txtHuffmanOriginal1.Text = txtHuffman1.Text;
        lblHuffmanOriginalMarker1.Text = lblHuffmanMarker1.Text;
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The txtHuffman2 TextBox object has generated a Click event.
    ///  Post-conditions:
    ///    If this is the first time the data has been altered, the data is
    ///    copied into the txtHuffmanOriginal2 TextBox.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    txtHuffman2 TextBox object.  The purpose of this TextBox is to
    ///    allow the user to manipulate the values in the second Huffman
    ///    table contained within the JPEG image.  If this is the first time
    ///    this data has been altered, this function copies the data from the
    ///    txtHuffman2 TextBox (before it has been changed) into the
    ///    txtHuffmanOriginal2 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void txtHuffman2_GotFocus(object sender, System.EventArgs e)
    {
      if(!LoadingInterface && this.txtHuffmanOriginal2.Text == "")
      {
        txtHuffmanOriginal2.Text = txtHuffman2.Text;
        lblHuffmanOriginalMarker2.Text = lblHuffmanMarker2.Text;
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The txtHuffman3 TextBox object has generated a Click event.
    ///  Post-conditions:
    ///    If this is the first time the data has been altered, the data is
    ///    copied into the txtHuffmanOriginal3 TextBox.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    txtHuffman3 TextBox object.  The purpose of this TextBox is to
    ///    allow the user to manipulate the values in the third Huffman table
    ///    contained within the JPEG image.  If this is the first time this
    ///    data has been altered, this function copies the data from the
    ///    txtHuffman3 TextBox (before it has been changed) into the
    ///    txtHuffmanOriginal3 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
```



```
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void txtHuffman3_GotFocus(object sender, System.EventArgs e)
    {
      if(!LoadingInterface && this.txtHuffmanOriginal3.Text == "")
      {
        txtHuffmanOriginal3.Text = txtHuffman3.Text;
        lblHuffmanOriginalMarker3.Text = lblHuffmanMarker3.Text;
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The txtHuffman4 TextBox object has generated a Click event.
    ///  Post-conditions:
    ///    If this is the first time the data has been altered, the data is
    ///    copied into the txtHuffmanOriginal4 TextBox.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    txtHuffman4 TextBox object.  The purpose of this TextBox is to
    ///    allow the user to manipulate the values in the fourth Huffman table
    ///    contained within the JPEG image.  If this is the first time this
    ///    data has been altered, this function copies the data from the
    ///    txtHuffman4 TextBox (before it has been changed) into the
    ///    txtHuffmanOriginal4 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void txtHuffman4_GotFocus(object sender, System.EventArgs e)
    {
      if(!LoadingInterface && this.txtHuffmanOriginal4.Text == "")
      {
        txtHuffmanOriginal4.Text = txtHuffman4.Text;
        lblHuffmanOriginalMarker4.Text = lblHuffmanMarker4.Text;
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The txtHuffman5 TextBox object has generated a Click event.
    ///  Post-conditions:
    ///    If this is the first time the data has been altered, the data is
    ///    copied into the txtHuffmanOriginal5 TextBox.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    txtHuffman5 TextBox object.  The purpose of this TextBox is to
    ///    allow the user to manipulate the values in the fifth Huffman table
    ///    contained within the JPEG image.  If this is the first time this
    ///    data has been altered, this function copies the data from the
    ///    txtHuffman5 TextBox (before it has been changed) into the
    ///    txtHuffmanOriginal5 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void txtHuffman5_GotFocus(object sender, System.EventArgs e)
    {
      if(!LoadingInterface && this.txtHuffmanOriginal5.Text == "")
      {
        txtHuffmanOriginal5.Text = txtHuffman5.Text;
        lblHuffmanOriginalMarker5.Text = lblHuffmanMarker5.Text;
      }
    }
```

```
    /// <summary>
    /// Pre-conditions:
    ///    The txtHuffman6 TextBox object has generated a Click event.
    ///  Post-conditions:
    ///    If this is the first time the data has been altered, the data is
    ///    copied into the txtHuffmanOriginal6 TextBox.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    txtHuffman6 TextBox object.  The purpose of this TextBox is to
    ///    allow the user to manipulate the values in the sixth Huffman table
    ///    contained within the JPEG image.  If this is the first time this
    ///    data has been altered, this function copies the data from the
    ///    txtHuffman6 TextBox (before it has been changed) into the
    ///    txtHuffmanOriginal6 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void txtHuffman6_GotFocus(object sender, System.EventArgs e)
    {
      if(!LoadingInterface && this.txtHuffmanOriginal6.Text == "")
      {
        txtHuffmanOriginal6.Text = txtHuffman6.Text;
        lblHuffmanOriginalMarker6.Text = lblHuffmanMarker6.Text;
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The txtHuffman7 TextBox object has generated a Click event.
    ///  Post-conditions:
    ///    If this is the first time the data has been altered, the data is
    ///    copied into the txtHuffmanOriginal7 TextBox.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    txtHuffman7 TextBox object.  The purpose of this TextBox is to
    ///    allow the user to manipulate the values in the seventh Huffman
    ///    table contained within the JPEG image.  If this is the first time
    ///    this data has been altered, this function copies the data from the
    ///    txtHuffman7 TextBox (before it has been changed) into the
    ///    txtHuffmanOriginal7 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void txtHuffman7_GotFocus(object sender, System.EventArgs e)
    {
      if(!LoadingInterface && this.txtHuffmanOriginal7.Text == "")
      {
        txtHuffmanOriginal7.Text = txtHuffman7.Text;
        lblHuffmanOriginalMarker7.Text = lblHuffmanMarker7.Text;
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The txtHuffman8 TextBox object has generated a Click event.
    ///  Post-conditions:
    ///    If this is the first time the data has been altered, the data is
    ///    copied into the txtHuffmanOriginal8 TextBox.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    txtHuffman8 TextBox object.  The purpose of this TextBox is to
    ///    allow the user to manipulate the values in the eighth Huffman table
```



```
    ///    contained within the JPEG image.  If this is the first time this
    ///    data has been altered, this function copies the data from the
    ///    txtHuffman8 TextBox (before it has been changed) into the
    ///    txtHuffmanOriginal8 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void txtHuffman8_GotFocus(object sender, System.EventArgs e)
    {
      if(!LoadingInterface && this.txtHuffmanOriginal8.Text == "")
      {
        txtHuffmanOriginal8.Text = txtHuffman8.Text;
        lblHuffmanOriginalMarker8.Text = lblHuffmanMarker8.Text;
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnRestoreQuantizer1 Button object has generated a Click
    ///    event.
    ///  Post-conditions:
    ///    The information stored within the txtQuantizerOriginal1 (the
    ///    original picture data) is copied back into the txtQuantizer1
    ///    TextBox object.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnRestoreQuantizer1 Button object.  The purpose of this Button
    ///    is to allow the user to restore the original data for this
    ///    Quantizer table to the txtQuantizer1 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnRestoreQuantizer1_Click(object sender, System.EventArgs e)
    {
      if(lblQuantizerOriginalMarker1.Text != "")
      {
        txtQuantizer1.Text = txtQuantizerOriginal1.Text;
        txtQuantizerOriginal1.Text = "";
        lblQuantizerMarker1.Text = lblQuantizerOriginalMarker1.Text;
        lblQuantizerOriginalMarker1.Text = "";
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnRestoreQuantizer2 Button object has generated a Click
    ///    event.
    ///  Post-conditions:
    ///    The information stored within the txtQuantizerOriginal2 (the
    ///    original picture data) is copied back into the txtQuantizer2
    ///    TextBox object.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnRestoreQuantizer2 Button object.  The purpose of this Button
    ///    is to allow the user to restore the original data for this
    ///    Quantizer table to the txtQuantizer2 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnRestoreQuantizer2_Click(object sender, System.EventArgs e)
    {
```

```
    if(lblQuantizerOriginalMarker2.Text != "")
    {
      txtQuantizer2.Text = txtQuantizerOriginal2.Text;
      txtQuantizerOriginal2.Text = "";
      lblQuantizerMarker2.Text = lblQuantizerOriginalMarker2.Text;
      lblQuantizerOriginalMarker2.Text = "";
    }
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnRestoreQuantizer3 Button object has generated a Click
  ///    event.
  ///  Post-conditions:
  ///    The information stored within the txtQuantizerOriginal3 (the
  ///    original picture data) is copied back into the txtQuantizer3
  ///    TextBox object.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnRestoreQuantizer3 Button object.  The purpose of this Button
  ///    is to allow the user to restore the original data for this
  ///    Quantizer table to the txtQuantizer3 TextBox.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnRestoreQuantizer3_Click(object sender, System.EventArgs e)
  {
    if(lblQuantizerOriginalMarker3.Text != "")
    {
      txtQuantizer3.Text = txtQuantizerOriginal3.Text;
      txtQuantizerOriginal3.Text = "";
      lblQuantizerMarker3.Text = lblQuantizerOriginalMarker3.Text;
      lblQuantizerOriginalMarker3.Text = "";
    }
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnRestoreQuantizer4 Button object has generated a Click
  ///    event.
  ///  Post-conditions:
  ///    The information stored within the txtQuantizerOriginal4 (the
  ///    original picture data) is copied back into the txtQuantizer4
  ///    TextBox object.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnRestoreQuantizer4 Button object.  The purpose of this Button
  ///    is to allow the user to restore the original data for this
  ///    Quantizer table to the txtQuantizer4 TextBox.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnRestoreQuantizer4_Click(object sender, System.EventArgs e)
  {
    if(lblQuantizerOriginalMarker4.Text != "")
    {
      txtQuantizer4.Text = txtQuantizerOriginal4.Text;
      txtQuantizerOriginal4.Text = "";
      lblQuantizerMarker4.Text = lblQuantizerOriginalMarker4.Text;
      lblQuantizerOriginalMarker4.Text = "";
    }
  }
```



```
  /// <summary>
  /// Pre-conditions:
  ///    The btnRestoreHuffman1 Button object has generated a Click event.
  ///  Post-conditions:
  ///    The information stored within the txtHuffmanOriginal1 (the
  ///    original picture data) is copied back into the txtHuffman1
  ///    TextBox object.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnRestoreHuffman1 Button object.  The purpose of this Button is
  ///    to allow the user to restore the original data for this Huffman
  ///    table to the txtHuffman1 TextBox.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnRestoreHuffman1_Click(object sender, System.EventArgs e)
  {
    if(lblHuffmanOriginalMarker1.Text != "")
    {
      txtHuffman1.Text = txtHuffmanOriginal1.Text;
      txtHuffmanOriginal1.Text = "";
      lblHuffmanMarker1.Text = lblHuffmanOriginalMarker1.Text;
      lblHuffmanOriginalMarker1.Text = "";
    }
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnRestoreHuffman2 Button object has generated a Click event.
  ///  Post-conditions:
  ///    The information stored within the txtHuffmanOriginal2 (the
  ///    original picture data) is copied back into the txtHuffman2
  ///    TextBox object.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnRestoreHuffman2 Button object.  The purpose of this Button is
  ///    to allow the user to restore the original data for this Huffman
  ///    table to the txtHuffman2 TextBox.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function. </param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnRestoreHuffman2_Click(object sender, System.EventArgs e)
  {
    if(lblHuffmanOriginalMarker2.Text != "")
    {
      txtHuffman2.Text = txtHuffmanOriginal2.Text;
      txtHuffmanOriginal2.Text = "";
      lblHuffmanMarker2.Text = lblHuffmanOriginalMarker2.Text;
      lblHuffmanOriginalMarker2.Text = "";
    }
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnRestoreHuffman3 Button object has generated a Click event.
  ///  Post-conditions:
  ///    The information stored within the txtHuffmanOriginal3 (the
  ///    original picture data) is copied back into the txtHuffman3
  ///    TextBox object.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnRestoreHuffman3 Button object.  The purpose of this Button is
```

```
    ///    to allow the user to restore the original data for this Huffman
    ///    table to the txtHuffman3 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnRestoreHuffman3_Click(object sender, System.EventArgs e)
    {
      if(lblHuffmanOriginalMarker3.Text != "")
      {
        txtHuffman3.Text = txtHuffmanOriginal3.Text;
        txtHuffmanOriginal3.Text = "";
        lblHuffmanMarker3.Text = lblHuffmanOriginalMarker3.Text;
        lblHuffmanOriginalMarker3.Text = "";
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnRestoreHuffman4 Button object has generated a Click event.
    ///  Post-conditions:
    ///    The information stored within the txtHuffmanOriginal4 (the
    ///    original picture data) is copied back into the txtHuffman4
    ///    TextBox object.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnRestoreHuffman4 Button object.  The purpose of this Button is
    ///    to allow the user to restore the original data for this Huffman
    ///    table to the txtHuffman4 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnRestoreHuffman4_Click(object sender, System.EventArgs e)
    {
      if(lblHuffmanOriginalMarker4.Text != "")
      {
        txtHuffman4.Text = txtHuffmanOriginal4.Text;
        txtHuffmanOriginal4.Text = "";
        lblHuffmanMarker4.Text = lblHuffmanOriginalMarker4.Text;
        lblHuffmanOriginalMarker4.Text = "";
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnRestoreHuffman5 Button object has generated a Click event.
    ///  Post-conditions:
    ///    The information stored within the txtHuffmanOriginal5 (the
    ///    original picture data) is copied back into the txtHuffman5
    ///    TextBox object.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnRestoreHuffman5 Button object.  The purpose of this Button is
    ///    to allow the user to restore the original data for this Huffman
    ///    table to the txtHuffman5 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnRestoreHuffman5_Click(object sender, System.EventArgs e)
    {
      if(lblHuffmanOriginalMarker5.Text != "")
      {
```



```
        txtHuffman5.Text = txtHuffmanOriginal5.Text;
        txtHuffmanOriginal5.Text = "";
        lblHuffmanMarker5.Text = lblHuffmanOriginalMarker5.Text;
        lblHuffmanOriginalMarker5.Text = "";
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnRestoreHuffman6 Button object has generated a Click event.
    ///  Post-conditions:
    ///    The information stored within the txtHuffmanOriginal6 (the
    ///    original picture data) is copied back into the txtHuffman6
    ///    TextBox object.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnRestoreHuffman7 Button object.  The purpose of this Button is
    ///    to allow the user to restore the original data for this Huffman
    ///    table to the txtHuffman7 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnRestoreHuffman6_Click(object sender, System.EventArgs e)
    {
      if(lblHuffmanOriginalMarker6.Text != "")
      {
        txtHuffman6.Text = txtHuffmanOriginal6.Text;
        txtHuffmanOriginal6.Text = "";
        lblHuffmanMarker6.Text = lblHuffmanOriginalMarker6.Text;
        lblHuffmanOriginalMarker6.Text = "";
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnRestoreHuffman7 Button object has generated a Click event.
    ///  Post-conditions:
    ///    The information stored within the txtHuffmanOriginal7 (the
    ///    original picture data) is copied back into the txtHuffman7
    ///    TextBox object.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnRestoreHuffman7 Button object.  The purpose of this Button is
    ///    to allow the user to restore the original data for this Huffman
    ///    table to the txtHuffman7 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnRestoreHuffman7_Click(object sender, System.EventArgs e)
    {
      if(lblHuffmanOriginalMarker7.Text != "")
      {
        txtHuffman7.Text = txtHuffmanOriginal7.Text;
        txtHuffmanOriginal7.Text = "";
        lblHuffmanMarker7.Text = lblHuffmanOriginalMarker7.Text;
        lblHuffmanOriginalMarker7.Text = "";
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnRestoreHuffman8 Button object has generated a Click event.
```

```
    ///  Post-conditions:
    ///    The information stored within the txtHuffmanOriginal8 (the
    ///    original picture data) is copied back into the txtHuffman8
    ///    TextBox object.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnRestoreHuffman8 Button object.  The purpose of this button is
    ///    to allow the user to restore the original data for this Huffman
    ///    table to the txtHuffman8 TextBox.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnRestoreHuffman8_Click(object sender, System.EventArgs e)
    {
      if(lblHuffmanOriginalMarker8.Text != "")
      {
        txtHuffman8.Text = txtHuffmanOriginal8.Text;
        txtHuffmanOriginal8.Text = "";
        lblHuffmanMarker8.Text = lblHuffmanOriginalMarker8.Text;
        lblHuffmanOriginalMarker8.Text = "";
      }
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnUpdate Menu Button object has generated a Click event.
    ///  Post-conditions:
    ///    A changed picture has been updated within the application.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnUpdate Menu Button object.  The purpose of this Button object
    ///    is to allow the user to create a new manipulated image for the
    ///    user to see.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnUpdate_Click(object sender, System.EventArgs e)
    {
      CreateISEImage();
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnNew Menu Button object has generated a Click event.
    ///  Post-conditions:
    ///    This function clears out all data for pictures.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnNew Menu Button object.  The purpose of this Button object is
    ///    to allow the user to create a new project file that will allow
    ///    them to store picture and note data about different images.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnNew_Click(object sender, System.EventArgs e)
    {
      ClearInterfaceData();
    }


    /// <summary>
```



```
    /// Pre-conditions:
    ///    The btnLoad Menu Button object has generated a Click event.
    ///  Post-conditions:
    ///    A previously created project file has been loaded by the
    ///    application.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnLoad Menu Button object.  The purpose of this Button object is
    ///    to allow the user to open a previously created project file.  The
    ///    values stored in the project file will be reloaded into the
    ///    application interface.  This function will simply call the
    ///    LoadNewProject() function described later in this document.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnLoad_Click(object sender, System.EventArgs e)
    {
      LoadNewProject();
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnSave Menu Button object has generated a Click event.
    ///  Post-conditions:
    ///    This function saves the current values loaded in the Manipulator
    ///    and any project notes, if included.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnSave Menu Button object.  The purpose of this Button object is
    ///    to allow the user to save a project file and all current
    ///    information in the application.  The values stored in the project
    ///    file will be reloaded into the application interface.  This
    ///    function will simply call the SaveNewProject() function described
    ///    later in this document.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnSave_Click(object sender, System.EventArgs e)
    {
      SaveNewProject();
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnLoadPicture Menu Button object has generated a Click event.
    ///  Post-conditions:
    ///    An image file has been loaded by the application.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnLoadPicture Menu Button object.  The purpose of this Button
    ///    object is to allow the user to open an image file.  The values
    ///    stored in the project file will be reloaded into the application
    ///    interface.  This function will simply call the LoadNewProject()
    ///    function described later in this document.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function. </param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnLoadPicture_Click(object sender, System.EventArgs e)
    {
      LoadNewPicture();
    }
```

```
/// <summary>
/// Pre-conditions:
///    The btnUpdatePicture Menu Button object has generated a Click
///    event.
///  Post-conditions:
///    A changed picture has been updated within the application.
///  Description:
///    This method is used to resolve a Click event generated by the
///    btnUpdatePicture Button object.  The purpose of this Button
///    object is to allow the user to create a manipulated image based
///    upon the data changed by user.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function. </param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void btnUpdatePicture_Click(object sender, System.EventArgs e)
{
  CreateISEImage();
}


/// <summary>
/// Pre-conditions:
///    The menuCut menu object has generated a Click event.
/// Post-conditions:
///    Selected text has been cut from the text box and copied to the
///    system clipboard.
///  Description:
///    This method is used to resolve a Click event generated by the
///    menuCut menu object.  The purpose of this menu object is to allow
///    the user to cut selected text from any TextBox field within the
///    Manipulator.  The cut text is copied to the system clipboard for
///    future retrieval.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void menuCopy_Click(object sender, System.EventArgs e)
{
  SendKeys.Send("^c");
}


/// <summary>
/// Pre-conditions:
///    The menuCopy menu object has generated a Click event.
///  Post-conditions:
///    Selected text has been copied to the system clipboard.
///  Description:
///    This method is used to resolve a Click event generated by the
///    menuCopy menu object.  The purpose of this menu object is to
///    allow the user to copy selected text from any TextBox field
///    within the Manipulator.  The text is copied to the system
///    clipboard for future retrieval.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void menuCut_Click(object sender, System.EventArgs e)
{
  SendKeys.Send("^x");
}
```



```
/// <summary>
/// Pre-conditions:
///    The menuPaste menu object has generated a Click event.
///  Post-conditions:
///    Most recent text on the system clipboard has been pasted to the
///    selected TextBox within the Manipulator.
///  Description:
///    This method is used to resolve a Click event generated by the
///    menuPaste menu object.  The purpose of this menu object is to
///    allow the user to copy the most recent text from the clipboard to
///    a selected Manipulator TextBox.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void menuPaste_Click(object sender, System.EventArgs e)
{
  SendKeys.Send("^v");
}


/// <summary>
/// Pre-conditions:
///    The btnClearHuffman1 button object has generated a Click event.
/// Post-conditions:
///    The corresponding txtHuffman1 text box has been cleared.
///  Description:
///    This method is used to resolve a Click event generated by the
///    btnClearHuffman1 button object.  The purpose of this button is to
///    allow the user to quickly clear out the corresponding txtHuffman1
///    text box control.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void btnClearHuffman1_Click(object sender, System.EventArgs e)
{
  if(txtHuffmanOriginal1.Text.Trim() == "")
  {
    txtHuffmanOriginal1.Text = txtHuffman1.Text;
    lblHuffmanOriginalMarker1.Text = lblHuffmanMarker1.Text;
  }

  txtHuffman1.Text = "";
}


/// <summary>
/// Pre-conditions:
///    The btnClearHuffman2 button object has generated a Click event.
/// Post-conditions:
///    The corresponding txtHuffman2 text box has been cleared.
///  Description:
///    This method is used to resolve a Click event generated by the
///    btnClearHuffman2 button object.  The purpose of this button is to
///    allow the user to quickly clear out the corresponding txtHuffman2
///    text box control.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void btnClearHuffman2_Click(object sender, System.EventArgs e)
{
  if(txtHuffmanOriginal2.Text.Trim() == "")
  {
    txtHuffmanOriginal2.Text = txtHuffman2.Text;
```

```
      lblHuffmanOriginalMarker2.Text = lblHuffmanMarker2.Text;
    }

    txtHuffman2.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearHuffman3 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtHuffman3 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearHuffman3 button object.  The purpose of this button is to
  ///    allow the user to quickly clear out the corresponding txtHuffman3
  ///    text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearHuffman3_Click(object sender, System.EventArgs e)
  {
    if(txtHuffmanOriginal3.Text.Trim() == "")
    {
      txtHuffmanOriginal3.Text = txtHuffman3.Text;
      lblHuffmanOriginalMarker3.Text = lblHuffmanMarker3.Text;
    }

    txtHuffman3.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearHuffman4 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtHuffman4 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearHuffman4 button object.  The purpose of this button is to
  ///    allow the user to quickly clear out the corresponding txtHuffman4
  ///    text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearHuffman4_Click(object sender, System.EventArgs e)
  {
    if(txtHuffmanOriginal4.Text.Trim() == "")
    {
      txtHuffmanOriginal4.Text = txtHuffman4.Text;
      lblHuffmanOriginalMarker4.Text = lblHuffmanMarker4.Text;
    }

    txtHuffman4.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearHuffman5 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtHuffman5 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearHuffman5 button object.  The purpose of this button is to
```



```
  ///    allow the user to quickly clear out the corresponding txtHuffman5
  ///    text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearHuffman5_Click(object sender, System.EventArgs e)
  {
    if(txtHuffmanOriginal5.Text.Trim() == "")
    {
      txtHuffmanOriginal5.Text = txtHuffman5.Text;
      lblHuffmanOriginalMarker5.Text = lblHuffmanMarker5.Text;
    }

    txtHuffman5.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearHuffman6 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtHuffman6 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearHuffman6 button object.  The purpose of this button is to
  ///    allow the user to quickly clear out the corresponding txtHuffman6
  ///    text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearHuffman6_Click(object sender, System.EventArgs e)
  {
    if(txtHuffmanOriginal6.Text.Trim() == "")
    {
      txtHuffmanOriginal6.Text = txtHuffman6.Text;
      lblHuffmanOriginalMarker6.Text = lblHuffmanMarker6.Text;
    }

    txtHuffman6.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearHuffman7 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtHuffman7 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearHuffman7 button object.  The purpose of this button is to
  ///    allow the user to quickly clear out the corresponding txtHuffman7
  ///    text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearHuffman7_Click(object sender, System.EventArgs e)
  {
    if(txtHuffmanOriginal7.Text.Trim() == "")
    {
      txtHuffmanOriginal7.Text = txtHuffman7.Text;
      lblHuffmanOriginalMarker7.Text = lblHuffmanMarker7.Text;
    }
```

```
    txtHuffman7.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearHuffman8 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtHuffman8 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearHuffman8 button object.  The purpose of this button is to
  ///    allow the user to quickly clear out the corresponding txtHuffman8
  ///    text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearHuffman8_Click(object sender, System.EventArgs e)
  {
    if(txtHuffmanOriginal8.Text.Trim() == "")
    {
      txtHuffmanOriginal8.Text = txtHuffman8.Text;
      lblHuffmanOriginalMarker8.Text = lblHuffmanMarker8.Text;
    }

    txtHuffman8.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearQuantizer1 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtQuantizer1 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearQuantizer1 button object.  The purpose of this button is
  ///    to allow the user to quickly clear out the corresponding
  ///    txtQuantizer1 text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearQuantizer1_Click(object sender, System.EventArgs e)
  {
    if(txtQuantizerOriginal1.Text.Trim()== "")
    {
      txtQuantizerOriginal1.Text = txtQuantizer1.Text;
      lblQuantizerOriginalMarker1.Text = lblQuantizerMarker1.Text;
    }

    txtQuantizer1.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearQuantizer2 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtQuantizer2 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearQuantizer2 button object.  The purpose of this button is
  ///    to allow the user to quickly clear out the corresponding
  ///    txtQuantizer2 text box control.
  /// </summary>
```



```
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearQuantizer2_Click(object sender, System.EventArgs e)
  {
    if(txtQuantizerOriginal2.Text.Trim() == "")
    {
      txtQuantizerOriginal2.Text = txtQuantizer2.Text;
      lblQuantizerOriginalMarker2.Text = lblQuantizerMarker2.Text;
    }

    txtQuantizer2.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearQuantizer3 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtQuantizer3 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearQuantizer3 button object.  The purpose of this button is
  ///    to allow the user to quickly clear out the corresponding
  ///    txtQuantizer3 text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearQuantizer3_Click(object sender, System.EventArgs e)
  {
    if(txtQuantizerOriginal3.Text.Trim() == "")
    {
      txtQuantizerOriginal3.Text = txtQuantizer3.Text;
      lblQuantizerOriginalMarker3.Text = lblQuantizerMarker3.Text;
    }

    txtQuantizer3.Text = "";
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnClearQuantizer4 button object has generated a Click event.
  /// Post-conditions:
  ///    The corresponding txtQuantizer4 text box has been cleared.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnClearQuantizer4 button object.  The purpose of this button is
  ///    to allow the user to quickly clear out the corresponding
  ///    txtQuantizer4 text box control.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnClearQuantizer4_Click(object sender, System.EventArgs e)
  {
    if(txtQuantizerOriginal4.Text.Trim() == "")
    {
      txtQuantizerOriginal4.Text = txtQuantizer4.Text;
      lblQuantizerOriginalMarker4.Text = lblQuantizerMarker4.Text;
    }

    txtQuantizer4.Text = "";
  }
```

```
/// <summary>
/// Pre-conditions:
///    The btnAddRandomHuffman1 button object has generated a Click
///    event.
/// Post-conditions:
///    The corresponding txtHuffman1 text box has a random byte
///    concatenated to the end of any text that was already existing in
///    the control.
///  Description:
///    This method is used to resolve a Click event generated by the
///    btnAddRandomHuffman1 button object.  The purpose of this button
///    is to allow the user to simulate adding a random byte to the end
///    of the existing text in the txtHuffman1 control. This data will
///    be represent the hexadecimal value of one byte of data.  In
///    addition this method will also add a space (" ") after the byte
///    of data.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void btnAddRandomHuffman1_Click(object sender, System.EventArgs e)
{
  int t = RandomNumber.Next(16);
  string a = Convert(t).ToString();

  if(txtHuffmanOriginal1.Text == "")
  {
    txtHuffmanOriginal1.Text = txtHuffman1.Text;
    lblHuffmanOriginalMarker1.Text = lblHuffmanMarker1.Text;
  }

  t = RandomNumber.Next(16);
  a += Convert(t).ToString() + " ";
  txtHuffman1.Text += a;
}


/// <summary>
/// Pre-conditions:
///    The btnAddRandomHuffman2 button object has generated a Click
///    event.
/// Post-conditions:
///    The corresponding txtHuffman2 text box has a random byte
///    concatenated to the end of any text that was already existing in
///    the control.
///  Description:
///    This method is used to resolve a Click event generated by the
///    btnAddRandomHuffman2 button object.  The purpose of this button is
///    to allow the user to simulate adding a random byte to the end of
///    the existing text in the txtHuffman2 control. This data will be
///    represent the hexadecimal value of one byte of data.  In addition
///    this method will also add a space (" ") after the byte of data.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void btnAddRandomHuffman2_Click(object sender, System.EventArgs e)
{
  int t = RandomNumber.Next(16);
  string a = Convert(t).ToString();

  if(txtHuffmanOriginal2.Text == "")
  {
    txtHuffmanOriginal2.Text = txtHuffman2.Text;
    lblHuffmanOriginalMarker2.Text = lblHuffmanMarker2.Text;
  }
```



```
  t = RandomNumber.Next(16);
  a += Convert(t).ToString() + " ";
  txtHuffman2.Text += a;
}


/// <summary>
/// Pre-conditions:
///    The btnAddRandomHuffman3 button object has generated a Click
///    event.
/// Post-conditions:
///    The corresponding txtHuffman3 text box has a random byte
///    concatenated to the end of any text that was already existing in
///    the control.
///  Description:
///    This method is used to resolve a Click event generated by the
///    btnAddRandomHuffman3 button object.  The purpose of this button is
///    to allow the user to simulate adding a random byte to the end of
///    the existing text in the txtHuffman3 control. This data will be
///    represent the hexadecimal value of one byte of data.  In addition
///    this method will also add a space (" ") after the byte of data.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void btnAddRandomHuffman3_Click(object sender, System.EventArgs e)
{
  int t = RandomNumber.Next(16);
  string a = Convert(t).ToString();

  if(txtHuffmanOriginal3.Text == "")
  {
    txtHuffmanOriginal3.Text = txtHuffman3.Text;
    lblHuffmanOriginalMarker3.Text = lblHuffmanMarker3.Text;
  }

  t = RandomNumber.Next(16);
  a += Convert(t).ToString() + " ";
  txtHuffman3.Text += a;
}


/// <summary>
/// Pre-conditions:
///    The btnAddRandomHuffman4 button object has generated a Click
///    event.
/// Post-conditions:
///    The corresponding txtHuffman4 text box has a random byte
///    concatenated to the end of any text that was already existing in
///    the control.
///  Description:
///    This method is used to resolve a Click event generated by the
///    btnAddRandomHuffman4 button object.  The purpose of this button is
///    to allow the user to simulate adding a random byte to the end of
///    the existing text in the txtHuffman4 control. This data will be
///    represent the hexadecimal value of one byte of data.  In addition
///    this method will also add a space (" ") after the byte of data.
/// </summary>
/// <param name="sender">The sender parameter is a reference to the
/// function calling this function.</param>
/// <param name="e">The e parameter is for the base class to pass event
/// data.</param>
private void btnAddRandomHuffman4_Click(object sender, System.EventArgs e)
{
  int t = RandomNumber.Next(16);
  string a = Convert(t).ToString();
```

```
    if(txtHuffmanOriginal4.Text == "")
    {
      txtHuffmanOriginal4.Text = txtHuffman4.Text;
      lblHuffmanOriginalMarker4.Text = lblHuffmanMarker4.Text;
    }

    t = RandomNumber.Next(16);
    a += Convert(t).ToString() + " ";
    txtHuffman4.Text += a;
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnAddRandomHuffman5 button object has generated a Click
  ///    event.
  /// Post-conditions:
  ///    The corresponding txtHuffman5 text box has a random byte
  ///    concatenated to the end of any text that was already existing in
  ///    the control.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnAddRandomHuffman5 button object.  The purpose of this button is
  ///    to allow the user to simulate adding a random byte to the end of
  ///    the existing text in the txtHuffman5 control. This data will be
  ///    represent the hexadecimal value of one byte of data.  In addition
  ///    this method will also add a space (" ") after the byte of data.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnAddRandomHuffman5_Click(object sender, System.EventArgs e)
  {
    int t = RandomNumber.Next(16);
    string a = Convert(t).ToString();

    if(txtHuffmanOriginal5.Text == "")
    {
      txtHuffmanOriginal5.Text = txtHuffman5.Text;
      lblHuffmanOriginalMarker5.Text = lblHuffmanMarker5.Text;
    }

    t = RandomNumber.Next(16);
    a += Convert(t).ToString() + " ";
    txtHuffman5.Text += a;
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnAddRandomHuffman6 button object has generated a Click
  ///    event.
  /// Post-conditions:
  ///    The corresponding txtHuffman6 text box has a random byte
  ///    concatenated to the end of any text that was already existing in
  ///    the control.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnAddRandomHuffman6 button object.  The purpose of this button is
  ///    to allow the user to simulate adding a random byte to the end of
  ///    the existing text in the txtHuffman6 control. This data will be
  ///    represent the hexadecimal value of one byte of data.  In addition
  ///    this method will also add a space (" ") after the byte of data.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
```



```
  private void btnAddRandomHuffman6_Click(object sender, System.EventArgs e)
  {
    int t = RandomNumber.Next(16);
    string a = Convert(t).ToString();

    if(txtHuffmanOriginal6.Text == "")
    {
      txtHuffmanOriginal6.Text = txtHuffman6.Text;
      lblHuffmanOriginalMarker6.Text = lblHuffmanMarker6.Text;
    }

    t = RandomNumber.Next(16);
    a += Convert(t).ToString() + " ";
    txtHuffman6.Text += a;
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnAddRandomHuffman7 button object has generated a Click
  ///    event.
  /// Post-conditions:
  ///    The corresponding txtHuffman7 text box has a random byte
  ///    concatenated to the end of any text that was already existing in
  ///    the control.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnAddRandomHuffman7 button object.  The purpose of this button is
  ///    to allow the user to simulate adding a random byte to the end of
  ///    the existing text in the txtHuffman7 control. This data will be
  ///    represent the hexadecimal value of one byte of data.  In addition
  ///    this method will also add a space (" ") after the byte of data.
  /// </summary>
  /// <param name="sender">The sender parameter is a reference to the
  /// function calling this function.</param>
  /// <param name="e">The e parameter is for the base class to pass event
  /// data.</param>
  private void btnAddRandomHuffman7_Click(object sender, System.EventArgs e)
  {
    int t = RandomNumber.Next(16);
    string a = Convert(t).ToString();

    if(txtHuffmanOriginal7.Text == "")
    {
      txtHuffmanOriginal7.Text = txtHuffman7.Text;
      lblHuffmanOriginalMarker7.Text = lblHuffmanMarker7.Text;
    }

    t = RandomNumber.Next(16);
    a += Convert(t).ToString() + " ";
    txtHuffman7.Text += a;
  }


  /// <summary>
  /// Pre-conditions:
  ///    The btnAddRandomHuffman8 button object has generated a Click
  ///    event.
  /// Post-conditions:
  ///    The corresponding txtHuffman8 text box has a random byte
  ///    concatenated to the end of any text that was already existing in
  ///    the control.
  ///  Description:
  ///    This method is used to resolve a Click event generated by the
  ///    btnAddRandomHuffman8 button object.  The purpose of this button is
  ///    to allow the user to simulate adding a random byte to the end of
  ///    the existing text in the txtHuffman8 control. This data will be
  ///    represent the hexadecimal value of one byte of data.  In addition
  ///    this method will also add a space (" ") after the byte of data.
```

```
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnAddRandomHuffman8_Click(object sender, System.EventArgs e)
    {
      int t = RandomNumber.Next(16);
      string a = Convert(t).ToString();

      if(txtHuffmanOriginal8.Text == "")
      {
        txtHuffmanOriginal8.Text = txtHuffman8.Text;
        lblHuffmanOriginalMarker8.Text = lblHuffmanMarker8.Text;
      }

      t = RandomNumber.Next(16);
      a += Convert(t).ToString() + " ";
      txtHuffman8.Text += a;
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnAddRandomQuantizer1 button object has generated a Click
    ///    event.
    /// Post-conditions:
    ///    The corresponding txtQuantizer1 text box has a random byte
    ///    concatenated to the end of any text that was already existing in
    ///    the control.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnAddRandomQuantizer1 button object.  The purpose of this button
    ///    is to allow the user to simulate adding a random byte to the end
    ///    of the existing text in the txtQuantizer1 control. This data will
    ///    be represent the hexadecimal value of one byte of data.  In
    ///    addition this method will also add a space (" ") after the byte
    ///    of data.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnAddRandomQuantizer1_Click(object sender, System.EventArgs e)
    {
      int t = RandomNumber.Next(16);
      string a = Convert(t).ToString();

      if(txtQuantizerOriginal1.Text == "")
      {
        txtQuantizerOriginal1.Text = txtQuantizer1.Text;
        lblQuantizerOriginalMarker1.Text = lblQuantizerMarker1.Text;
      }

      t = RandomNumber.Next(16);
      a += Convert(t).ToString() + " ";
      txtQuantizer1.Text += a;
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnAddRandomQuantizer2 button object has generated a Click
    ///    event.
    /// Post-conditions:
    ///    The corresponding txtQuantizer2 text box has a random byte
    ///    concatenated to the end of any text that was already existing in
    ///    the control.
    ///  Description:
```



```
    ///    This method is used to resolve a Click event generated by the
    ///    btnAddRandomQuantizer2 button object.  The purpose of this button
    ///    is to allow the user to simulate adding a random byte to the end
    ///    of the existing text in the txtQuantizer2 control. This data will
    ///    be represent the hexadecimal value of one byte of data.  In
    ///    addition this method will also add a space (" ") after the byte
    ///    of data.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnAddRandomQuantizer2_Click(object sender,
                System.EventArgs e)
    {
      int t = RandomNumber.Next(16);
      string a = Convert(t).ToString();

      if(txtQuantizerOriginal2.Text == "")
      {
        txtQuantizerOriginal2.Text = txtQuantizer2.Text;
        lblQuantizerOriginalMarker2.Text = lblQuantizerMarker2.Text;
      }

      t = RandomNumber.Next(16);
      a += Convert(t).ToString() + " ";
      txtQuantizer2.Text += a;
    }


    /// <summary>
    /// Pre-conditions:
    ///    The btnAddRandomQuantizer3 button object has generated a Click
    ///    event.
    /// Post-conditions:
    ///    The corresponding txtQuantizer3 text box has a random byte
    ///    concatenated to the end of any text that was already existing in
    ///    the control.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnAddRandomQuantizer3 button object.  The purpose of this button
    ///    is to allow the user to simulate adding a random byte to the end
    ///    of the existing text in the txtQuantizer3 control. This data will
    ///    be represent the hexadecimal value of one byte of data.  In
    ///    addition this method will also add a space (" ") after the byte
    ///    of data.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnAddRandomQuantizer3_Click(object sender,
                System.EventArgs e)
    {
      int t = RandomNumber.Next(16);
      string a = Convert(t).ToString();

      if(txtQuantizerOriginal3.Text == "")
      {
        txtQuantizerOriginal3.Text = txtQuantizer3.Text;
        lblQuantizerOriginalMarker3.Text = lblQuantizerMarker3.Text;
      }

      t = RandomNumber.Next(16);
      a += Convert(t).ToString() + " ";
      txtQuantizer3.Text += a;
    }
```

```
    /// <summary>
    /// Pre-conditions:
    ///    The btnAddRandomQuantizer4 button object has generated a Click
    ///    event.
    /// Post-conditions:
    ///    The corresponding txtQuantizer4 text box has a random byte
    ///    concatenated to the end of any text that was already existing in
    ///    the control.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    btnAddRandomQuantizer4 button object.  The purpose of this button
    ///    is to allow the user to simulate adding a random byte to the end
    ///    of the existing text in the txtQuantizer4 control. This data will
        ///    be represent the hexadecimal value of one byte of data.  In
        ///    addition this method will also add a space (" ") after the byte
        ///    of data.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void btnAddRandomQuantizer4_Click(object sender,
                System.EventArgs e)
    {
      int t = RandomNumber.Next(16);
      string a = Convert(t).ToString();

      if(txtQuantizerOriginal4.Text == "")
      {
        txtQuantizerOriginal4.Text = txtQuantizer4.Text;
        lblQuantizerOriginalMarker4.Text = lblQuantizerMarker4.Text;
      }

      t = RandomNumber.Next(16);
      a += Convert(t).ToString() + " ";
      txtQuantizer4.Text += a;
    }


    /// <summary>
    /// Pre-conditions:
    ///    The menuUpdate Menu object has generated a Click event.
    ///  Post-conditions:
    ///    A changed picture has been updated within the application.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    menuUpdate Menu object.  The purpose of this Menu object is to
    ///    allow the user to create a manipulated image based upon the data
    ///    changed by user.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void menuUpdate_Click(object sender, System.EventArgs e)
    {
      CreateISEImage();
    }


    /// <summary>
    /// Pre-conditions:
    ///    The menuLargeOriginal Menu object has generated a Click event.
    ///  Post-conditions:
    ///    If the picture in the picOriginal is in "normal" size mode, it
    ///    will be changed to "stretch" size mode, otherwise it will be
    ///    switched to "normal" size mode.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
```



```
    ///    menuLargeOriginal Menu object.  The purpose of this Menu object is
    ///    to allow the user to toggle between "normal" and "stretch" size
    ///    modes for the original picture on the tabOriginal Tab control.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void menuLargeOriginal_Click(object sender, System.EventArgs e)
    {
      if(PicOriginalStretched)
      {
        PicOriginalStretched = false;
        menuLargeOriginal.Checked = false;
        picOriginal.SizeMode = PictureBoxSizeMode.Normal;
        menuAll.Checked = false;
      }
      else
      {
        PicOriginalStretched = true;
        menuLargeOriginal.Checked = true;
        picOriginal.SizeMode = PictureBoxSizeMode.StretchImage;
        if(menuSmallManipulated.Checked == true &&
          menuSmallOriginal.Checked == true &&
          menuLargeManipulated.Checked == true)
        {
          menuAll.Checked = true;
        }
      }
      picOriginal.Update();
    }


    /// <summary>
    /// Pre-conditions:
    ///    The menuLargeManipulated Menu object has generated a Click event.
    ///  Post-conditions:
    ///    If the picture in the picManipulated is in "normal" size mode, it
    ///    will be changed to "stretch" size mode, otherwise it will be
    ///    switched to "normal" size mode.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    menuLargeManipulated Menu object.  The purpose of this Menu object is

    ///    to allow the user to toggle between "normal" and "stretch" size
    ///    for the changed picture on the tabManipulated Tab control.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void menuLargeManipulated_Click(object sender, System.EventArgs e)
    {
      if(PicManipulatedStretched)
      {
        PicManipulatedStretched = false;
        menuLargeManipulated.Checked = false;
        picManipulated.SizeMode = PictureBoxSizeMode.Normal;
        menuAll.Checked = false;
      }
      else
      {
        PicManipulatedStretched = true;
        menuLargeManipulated.Checked = true;
        picManipulated.SizeMode = PictureBoxSizeMode.StretchImage;
        if(menuSmallManipulated.Checked == true &&
          menuSmallOriginal.Checked == true &&
          menuLargeOriginal.Checked == true)
        {
```

```
          menuAll.Checked = true;
        }
      }
      picManipulated.Update();
    }


    /// <summary>
    /// Pre-conditions:
    ///    The menuSmallOriginal Menu object has generated a Click event.
    ///  Post-conditions:
    ///    If the picture in the picOriginalSmall is in "normal" size mode,
    ///    it will be changed to "stretch" size mode, otherwise it will be
    ///    switched to "normal" size mode.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    menuSmallOriginal Menu object.  The purpose of this Menu object is
    ///    to allow the user to toggle between "normal" and "stretch" size
    ///    modes for the original picture on the tabConsole Tab control.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void menuSmallOriginal_Click(object sender, System.EventArgs e)
    {
      if(PicOriginalSmallStretched)
      {
        PicOriginalSmallStretched = false;
        menuSmallOriginal.Checked = false;
        picOriginalSmall.SizeMode = PictureBoxSizeMode.Normal;
        menuAll.Checked = false;
      }
      else
      {
        PicOriginalSmallStretched = true;
        menuSmallOriginal.Checked = true;
        picOriginalSmall.SizeMode = PictureBoxSizeMode.StretchImage;
        if(menuSmallManipulated.Checked == true &&
          menuLargeManipulated.Checked == true &&
          menuLargeOriginal.Checked == true)
        {
          menuAll.Checked = true;
        }
      }
      picOriginalSmall.Update();
    }


    /// <summary>
    /// Pre-conditions:
    ///    The menuSmallManipulated Menu object has generated a Click event.
    ///  Post-conditions:
    ///    If the picture in the picManipulatedSmall is in "normal" size mode, i
t
    ///    will be changed to "stretch" size mode, otherwise it will be
    ///    switched to "normal" size mode.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    menuSmallManipulated Menu object.  The purpose of this Menu object is

    ///    to allow the user to toggle between "normal" and "stretch" size
    ///    modes for the original picture on the tabConsole Tab control.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void menuSmallManipulated_Click(object sender, System.EventArgs e)
```



```
    {
      if(PicManipulatedSmallStretched)
      {
        PicManipulatedSmallStretched = false;
        menuSmallManipulated.Checked = false;
        picManipulatedSmall.SizeMode = PictureBoxSizeMode.Normal;
        menuAll.Checked = false;
      }
      else
      {
        PicManipulatedSmallStretched = true;
        menuSmallManipulated.Checked = true;
        picManipulatedSmall.SizeMode = PictureBoxSizeMode.StretchImage;
        if(menuSmallOriginal.Checked == true &&
          menuLargeManipulated.Checked == true &&
          menuLargeOriginal.Checked == true)
        {
          menuAll.Checked = true;
        }
      }
      picManipulatedSmall.Update();
    }


    /// <summary>
    /// Pre-conditions:
    ///    The menuAll Menu object has generated a Click event.
    ///  Post-conditions:
    ///    The menuAll Menu control will become selected and all pictures
    ///    will be switched to "stretch" size mode.  If this menu has been
    ///    previously selected, all of the pictures will be switched to
    ///    "normal" size mode instead.
    ///  Description:
    ///    This method is used to resolve a Click event generated by the
    ///    menuAll Menu object.  The purpose of this Menu object is to
    ///    allow the user to toggle between "normal" and "stretch" size
    ///    modes for the all of the pictures on the all of the Tab control.
    /// </summary>
    /// <param name="sender">The sender parameter is a reference to the
    /// function calling this function.</param>
    /// <param name="e">The e parameter is for the base class to pass event
    /// data.</param>
    private void menuAll_Click(object sender, System.EventArgs e)
    {
      if(menuAll.Checked)
      {
        menuAll.Checked = false;
        PicOriginalStretched = false;
        menuLargeOriginal.Checked = false;
        picOriginal.SizeMode = PictureBoxSizeMode.Normal;
        PicManipulatedStretched = false;
        menuLargeManipulated.Checked = false;
        picManipulated.SizeMode = PictureBoxSizeMode.Normal;
        PicOriginalSmallStretched = false;
        menuSmallOriginal.Checked = false;
        picOriginalSmall.SizeMode = PictureBoxSizeMode.Normal;
        PicManipulatedSmallStretched = false;
        menuSmallManipulated.Checked = false;
        picManipulatedSmall.SizeMode = PictureBoxSizeMode.Normal;
      }
      else
      {
        menuAll.Checked = true;
        PicOriginalStretched = true;
        menuLargeOriginal.Checked = true;
        picOriginal.SizeMode = PictureBoxSizeMode.StretchImage;
        PicManipulatedStretched = true;
        menuLargeManipulated.Checked = true;
        picManipulated.SizeMode = PictureBoxSizeMode.StretchImage;
```

```
      PicOriginalSmallStretched = true;
      menuSmallOriginal.Checked = true;
      picOriginalSmall.SizeMode = PictureBoxSizeMode.StretchImage;
      PicManipulatedSmallStretched = true;
      menuSmallManipulated.Checked = true;
      picManipulatedSmall.SizeMode = PictureBoxSizeMode.StretchImage;
    }
    picOriginal.Update();
    picManipulated.Update();
    picOriginalSmall.Update();
    picManipulatedSmall.Update();
  }


  private void frmMain_Load(object sender, System.EventArgs e)
  {
    // Create the new splash screen
    SplashScreen = new frmSplash();
    SplashScreen.Show();

    // Set the timer
    timerSplash.Enabled = true;
    timerSplash.Interval = 2000; // 2000 millisecs = 2 secs
    timerSplash.Start();
  }


  private void timerSplash_Tick(object sender, System.EventArgs e)
  {
    // Close the splash screen once the timer expires
    SplashScreen.Close();
    SplashScreen.Dispose();
    timerSplash.Dispose();
  }

  private void menuTutorial_Click(object sender, System.EventArgs e)
  {
    System.Windows.Forms.Help.ShowHelp(
      this, ProgramDirectory + @"\ISE Manipulator Tutorial.pdf");
  }


  private void menuManual_Click(object sender, System.EventArgs e)
  {
    System.Windows.Forms.Help.ShowHelp(
      this, ProgramDirectory + @"\ISE Manipulator Manual.pdf");
  }



  #endregion Interface Methods


  #region Common Methods

  /// <summary>
  /// Pre-conditions:    None.
  /// Post-conditions:
  ///    An original JPEG image has been loaded into the picOriginal and
  ///    picOriginalSmall PictureBox data members and a manipulated JPEG
  ///    image has been loaded into the picManipulated and
  ///    picManipulatedSmall data members.  Also, all of the data contained
  ///    in the original file should be loaded into the interface to
  ///    display for the user.
  ///  Description:
  ///    This method should be called if the Manipulator needs to be
  ///    completely reload. This method should be used by any other function
  ///    that needs to reload both images and the data into the interface.
  ///    This method should check to make sure that any previous image has
```



```
  ///    been closed within the picOriginal, picOriginalSmall,
  ///    picManipulated and picManipulatedSmall PictureBox controls before
  ///    trying to load the new images.  This function should do some error
  ///    checking to make sure that these files actually exist before
  ///    trying to load them. If one (or both) of the parameters does not
  ///    contain a valid file name and path, then it should be ignored and
  ///    an error message should be displayed in the txtError.  If an image
  ///    exists, yet it is too far damaged to load into the PictureBox
  ///    controls, then an error message should be displayed for the user
  ///    to see.  If any errors occur during load time, the error should
  ///    be displayed in the txtError TextBox for the user to see.
  ///
  ///    To perform this functionality, this function should call
  ///    ClearInterfaceData(), to clear the interface.  It should call
  ///    UpdateManipulatedPicture() to load the picManipulated picture. If
  ///    a valid file doesnM-^Rt exist in the ManipulatedFilePath parameter,
  ///    then it should just load the file in the OriginalFilePath
  ///    parameter.  If the OriginalFilePath parameter doesnM-^Rt contain a
  ///    valid file, this function should call one of the ShowWarning()
  ///    methods to let the user know that the OriginalFilePath is an
  ///    invalid file and in that case, no data should be loaded to the
  ///    interface.  This function should set the txtOrginalFile data
  ///    member.  It should also open the original file in the picOriginal
  ///    and picOriginalSmall PictureBox data members.  Lastly, this
  ///    function should call LoadPictureData() for the original file to
  ///    load all of the data into the TextBox fields of the Manipulator.
  /// </summary>
  /// <param name="OriginalFilePath">The OriginalFilePath parameter is a
  /// file path of the to the image to be loaded into the picOriginal and
  /// picOriginalSmall.</param>
  /// <param name="ManipulatedFilePath">The ManipulatedFilePath parameter
  /// is a file path of the to the image to be loaded into the
  /// picManipulated and picManipulatedSmall.</param>
  private void LoadPicture(string OriginalFilePath,
              string ManipulatedFilePath)
  {
    // To solve the problem with controls not losing focus when
    // a new picture is loaded.
    this.tabFile.Focus();
    this.Update();

    try
    {
      LoadingInterface = true;

      if(txtOriginalFile.Text != "")
      {
        if(!ShowWarning(
          "\nYou currently have a file open for editing.\n" +
          "If you open a newfile, all unsaved data will be lost!\n" +
          "Are you sure you want to open this new file?"))
        {
          LoadingInterface = false;
          return;
        }
        ClearInterfaceData();

      } // End of: if(txtOriginalFile.Text != "")

      this.Update();

      // This is for the Original Picture
      // Clear out the old image
      if(JPEG != null) JPEG.Dispose();
      if(JPEGsmall != null) JPEGsmall.Dispose();

      // Load the Original pic and resize to control size.
      JPEG = new Bitmap(OriginalFilePath);
      if(menuLargeOriginal.Checked)
```

```
      {
        PicOriginalStretched = true;
        picOriginal.SizeMode = PictureBoxSizeMode.StretchImage;
      }
      else
      {
        PicOriginalStretched = false;
        picOriginal.SizeMode = PictureBoxSizeMode.Normal;
      }
      picOriginal.Image = (Image)JPEG;
      picOriginal.Update();

      // Load the console tab picture too
      JPEGsmall = new Bitmap(OriginalFilePath);
      if(menuSmallOriginal.Checked)
      {
        PicOriginalSmallStretched = true;
        picOriginalSmall.SizeMode = PictureBoxSizeMode.StretchImage;
      }
      else
      {
        PicOriginalSmallStretched = false;
        picOriginalSmall.SizeMode = PictureBoxSizeMode.Normal;
      }
      picOriginalSmall.Image = (Image)JPEGsmall;
      picOriginalSmall.Update();

      // Load the Manipulated pic from same picture.
      UpdateManipulatedPicture(ManipulatedFilePath);

      // Update File Info
      txtOriginalFile.Text = OriginalFilePath;

      // Create a name for the changed file
      ManipulatedFileName = ManipulatedFilePath;
      txtManipulatedFile.Text = ManipulatedFileName;
      this.Update();

      // Load all of the Data Values into the interface
      LoadPictureData(OriginalFilePath);

      // Update frmMain Text
      this.Text = "ISE JPEG Manipulator - Version " + VERSION + "  -  "
        + openFileDialog.FileName;

      LoadingInterface = false;

    } // End of: try block
    catch(Exception ex)
    {
      if(ex.Message == "Invalid parameter used." ||
        ex.Message == "A generic error occurred in GDI+."||
        ex.Source == "System.Drawing")
      {
        OriginalFilePath = ProgramDirectory + @"\default_bad.jpg";
        LoadPicture(OriginalFilePath, OriginalFilePath);
      }
      else
      {
        ShowWarning(
          "Warning, an exception occured:\n\n" +
          "Exception Error:\n" +
          ex.Message + "\n\nWas throw by:\n" +
          ex.Source +
          "\n\nNot all load operations completed.!",
          "Load File Exception");
        ClearInterfaceData();
      }
    }
```



```
    LoadingInterface = false;

  } // End of: private void LoadPicture()


  /// <summary>
  /// See previous method definition.
  /// </summary>
  private void LoadNewPicture()
  {
    try
    {
      LoadingInterface = true;

      if(txtOriginalFile.Text != "")
      {

        if(!ShowWarning(
          "\nYou currently have a file open for editing.\n" +
          "If you open a newfile, all unsaved data will be lost!\n" +
          "Are you sure you want to open this new file?"))
        {
          LoadingInterface = false;
          return;
        }
      } // End of: if(txtOriginalFile.Text != "")

      else if(txtProjectPath.Text != "")
      {
        if(!ShowWarning(
          "\nYou currently have a file open for editing.\n" +
          "If you open a newfile, all unsaved data will be lost!\n" +
          "Are you sure you want to open this new file?"))
        {
          LoadingInterface = false;
          return;
        }
      } // End of: if(txtOriginalFile.Text != "")

      ClearInterfaceData();

      openFileDialog.ShowHelp = false;
      if(openFileDialog.ShowDialog() == DialogResult.OK)
      {
        this.Update();

        // This is for the Original Picture
        // Clear out the old image
        if(JPEG != null) JPEG.Dispose();
        if(JPEGsmall != null) JPEGsmall.Dispose();

        // Load the Original pic and resize to control size.
        JPEG = new Bitmap(openFileDialog.FileName);
        if(menuLargeOriginal.Checked)
        {
          PicOriginalStretched = true;
          picOriginal.SizeMode = PictureBoxSizeMode.StretchImage;
        }
        else
        {
          PicOriginalStretched = false;
          picOriginal.SizeMode = PictureBoxSizeMode.Normal;
        }
        picOriginal.Image = (Image)JPEG;
        picOriginal.Update();

        // Load the console tab picture too
        JPEGsmall = new Bitmap(openFileDialog.FileName);
        if(menuSmallOriginal.Checked)
```

```
          {
            PicOriginalSmallStretched = true;
            picOriginalSmall.SizeMode = PictureBoxSizeMode.StretchImage;
          }
          else
          {
            PicOriginalSmallStretched = false;
            picOriginalSmall.SizeMode = PictureBoxSizeMode.Normal;
          }
          picOriginalSmall.Image = (Image)JPEGsmall;
          picOriginalSmall.Update();

          // Load the Manipulated pic from same picture.
          UpdateManipulatedPicture(openFileDialog.FileName);

          // Update File Info
          txtOriginalFile.Text = openFileDialog.FileName;

          // Create a name for the changed file
          ManipulatedFileName = openFileDialog.FileName;
          string ttt = ManipulatedFileName.ToLower();
          ManipulatedFileName = ManipulatedFileName.ToLower();
          Count = ttt.IndexOf(".jpg");

          // Manipulated the file name if it already exists
          ManipulatedFileName =
                           ManipulatedFileName.Insert(Count, "_changed0");
          Temp = 0;
          string num_length;
          while(File.Exists(ManipulatedFileName))
          {
            Count = ManipulatedFileName.IndexOf(Temp.ToString() + ".jpg");
            num_length = Temp.ToString();
            ManipulatedFileName =
                              ManipulatedFileName.Remove(Count, num_length.Len
gth);
            Temp++;
            ManipulatedFileName =
                              ManipulatedFileName.Insert(Count, Temp.ToString(
));
          }

          txtManipulatedFile.Text = ManipulatedFileName;
          this.Update();

          // Load all of the Data Values
          LoadPictureData(openFileDialog.FileName);

          // Update frmMain Text
          this.Text = "ISE JPEG Manipulator - Version " + VERSION + "  -  "
            + openFileDialog.FileName;

          LoadingInterface = false;
        }

      } // End of: try block
      catch(Exception ex)
      {
        if(ex.Message == "Invalid parameter used." ||
          ex.Message == "A generic error occurred in GDI+." ||
          ex.Source == "System.Drawing")
        {
          string x = ProgramDirectory + @"\default_bad.jpg";
          LoadPicture(x, x);
        }
        else
        {
          ShowWarning(
            "Warning, an exception occured:\n\n" +
```



```
            "Exception Error:\n" +
            ex.Message + "\n\nWas throw by:\n" +
            ex.Source +
            "\n\nNot all load operations completed.!",
            "Load File Exception");
          ClearInterfaceData();
        }
      }
      LoadingInterface = false;

    } // End of: private void LoadNewPicture()


    /// <summary>
    /// Pre-conditions:
    ///    The data of an image has been previously loaded into the
    ///    Manipulator.
    ///  Post-conditions:
    ///    A new image based on the FileName parameter has been loaded into
    ///    the picManipulated and the picManipulatedSmall data fields.
    ///  Description:
    ///    This function is used to update picManipulated and
    ///    picManipulatedSmall data members, by loading a pre-existing
    ///    image.  If the FileName parameter is not a valid JPEG image, then
    ///    an error message should be displayed by calling the ShowWarning()
    ///    method.  Lastly, this method should do some error checking to
    ///    make sure this function executes properly. If an error is
    ///    encountered, then the ShowWarning() method should be called to
    ///    display the error to the user and the txtError TextBox control
    ///    should be updated with this error information.
    /// </summary>
    /// <param name="FileName">The FileName parameter is the name and path
    /// of a JPEG file to be loaded.</param>
    private void UpdateManipulatedPicture(string FileName)
    {
      try
      {
        // This is for the Manipulated Picture
        //
        // Clear out the old images
        if(ISE != null) ISE.Dispose();
        if(ISEsmall != null) ISEsmall.Dispose();

        // Open the new file and resize to control size.
        ISE = new Bitmap(FileName);
        if(menuLargeManipulated.Checked)
        {
          PicManipulatedStretched = true;
          picManipulated.SizeMode = PictureBoxSizeMode.StretchImage;
        }
        else
        {
          PicManipulatedStretched = false;
          picManipulated.SizeMode = PictureBoxSizeMode.Normal;
        }
        picManipulated.Image = (Image)ISE;
        picManipulated.Update();

        // Load the console tab picture too
        ISEsmall = new Bitmap(FileName);
        if(menuSmallManipulated.Checked)
        {
          PicManipulatedSmallStretched = true;
          picManipulatedSmall.SizeMode = PictureBoxSizeMode.StretchImage;
        }
        else
        {
          PicManipulatedSmallStretched = false;
          picManipulatedSmall.SizeMode = PictureBoxSizeMode.Normal;
```

```
        }
        picManipulatedSmall.Image = (Image)ISEsmall;
        picManipulatedSmall.Update();
      }
      catch(Exception ex)
      {
        if(ex.Message == "Invalid parameter used." ||
          ex.Message == "A generic error occurred in GDI+." ||
          ex.Source == "System.Drawing")
        {
          UpdateManipulatedPicture(ProgramDirectory + @"\default_bad.jpg");
        }
        else
        {
          if(ShowWarning(
            "An Exception Occured!" +
                        "\n\nThe Manipulator Failed to Load the File properly."
+
            "\n\nException Message: " + ex.Message + "\n\n" + ex.ToString() +
            "\n\nDo you want to reload the original picture?",
            "An Exception Occured!"
            ))
          {
            UpdateManipulatedPicture(txtOriginalFile.Text.Trim());
          }
          else
          {
            UpdateManipulatedPicture(
              ProgramDirectory + @"\default_bad.jpg");
          }
        }
      }
    }  // End of: private void UpdateManipulatedPicture(string FileName)


    /// <summary>
    /// Pre-conditions:    None.
    /// Post-conditions:
    ///    A warning message box is displayed for the user to see and decide
    ///    how to proceed.  This box will be shown until the user clicks
    ///    either the Ok or Cancel Button control on this message box, at
    ///    which point, this method will exit.
    ///  Description:
    ///    The purpose of this method is to be used by any method that wants
    ///    to display a warning message to the user.  In addition, this
    ///    method should return a True or False value, depending on the
    ///    response given by the user receiving this message.  This method
    ///    should call the standard MessageBox control to show the message.
    /// </summary>
    /// <param name="message">The message parameter is explanation of the
    /// warning message.</param>
    /// <param name="caption">The caption parameter is Window title of
    /// warning message box.</param>
    /// <returns>Function returns True if the user has clicked Ok and False
    /// if the user has clicked Cancel. </returns>
    private bool ShowWarning(string message, string caption)
    {
      string t = message.ToString();
      if(!(t.Length > 0)) t = "";
      if(MessageBox.Show(
        "Warning:\n" + t,
        caption,
        MessageBoxButtons.OKCancel,
        MessageBoxIcon.Error) == DialogResult.OK)
      {
        return true;
      }
      else return false;
    }
```



```


    /// <summary>
    /// Pre-conditions:    None.
    /// Post-conditions:
    ///    A warning message box is displayed for the user to see and decide
    ///    how to proceed.  This box will be shown until the user clicks
    ///    either the Ok or Cancel Button control on this message box, at
    ///    which point, this method will exit.
    ///  Description:
    ///    This function is a simpler version of the other ShowWarning
    ///    method. This function will create a default title for the warning
    ///    message box.  Then, this function will call the other
    ///    ShowWarning(string message, string caption) method with the
    ///    message parameter and the default title created.
    /// </summary>
    /// <param name="message">The message parameter is explanation of the
    /// warning message.</param>
    /// <returns>Function returns True if the user has clicked Ok and False
    /// if the user has clicked Cancel.</returns>
    private bool ShowWarning(string message)
    {
      return ShowWarning(message, "Warning!");
    }


    /// <summary>
    /// Pre-conditions:    None.
    /// Post-conditions:
    ///    All of the TextBox controls for all of the data fields within the
    ///    Manipulator will be reinitialized to empty strings.
    ///  Description:
    ///    This purpose of this method is to be called by any other method
    ///    that needs to clear out all of the data fields within the user
    ///    interface.  Specifically, this method should set all of the
    ///    strings to empty in every TextBox control found in the data
    ///    sub-tabs of the Console tab on the Manipulator frmMain Form.
    ///    It should also clear out all of the PictureBox controls within
    ///    all of the Tab controls of the application.
    /// </summary>
    private void ClearInterfaceData()
    {
      // Text fields to clear.
      this.txtApplicationData1.Text = "";
      this.txtApplicationData2.Text = "";
      this.txtApplicationData3.Text = "";
      this.txtApplicationData4.Text = "";
      this.txtApplicationData5.Text = "";
      this.txtApplicationData6.Text = "";
      this.txtApplicationData7.Text = "";
      this.txtApplicationData8.Text = "";
      this.txtApplicationData9.Text = "";
      this.txtApplicationData10.Text = "";

      this.txtManipulatedFile.Text = "";
      this.txtComments.Text = "";
      this.txtEncodedData.Text = "";
      this.txtError.Text = "";
      this.txtExpand.Text = "";
      this.txtFileSize.Text = "0";
      this.txtHierarchial.Text = "";
      this.txtNumberLines.Text = "";
      this.txtOriginalEncodedData.Text = "";
      this.txtOriginalFile.Text = "";
      this.txtOriginalHeader.Text = "";
      this.txtRestart.Text = "";
      this.txtRestartMod8.Text = "";
      this.txtScanHeader.Text = "";
```

```
      this.txtHuffman1.Text = "";
      this.txtHuffman2.Text = "";
      this.txtHuffman3.Text = "";
      this.txtHuffman4.Text = "";
      this.txtHuffman5.Text = "";
      this.txtHuffman6.Text = "";
      this.txtHuffman7.Text = "";
      this.txtHuffman8.Text = "";
      this.txtHuffmanOriginal1.Text = "";
      this.txtHuffmanOriginal2.Text = "";
      this.txtHuffmanOriginal3.Text = "";
      this.txtHuffmanOriginal4.Text = "";
      this.txtHuffmanOriginal5.Text = "";
      this.txtHuffmanOriginal6.Text = "";
      this.txtHuffmanOriginal7.Text = "";
      this.txtHuffmanOriginal8.Text = "";

      this.txtQuantizer1.Text = "";
      this.txtQuantizer2.Text = "";
      this.txtQuantizer3.Text = "";
      this.txtQuantizer4.Text = "";
      this.txtQuantizerOriginal1.Text = "";
      this.txtQuantizerOriginal2.Text = "";
      this.txtQuantizerOriginal3.Text = "";
      this.txtQuantizerOriginal4.Text = "";
      this.txtQuantizerTableNum1.Text = "";
      this.txtQuantizerTableNum2.Text = "";
      this.txtQuantizerTableNum3.Text = "";
      this.txtQuantizerTableNum4.Text = "";

      this.txtProjectPath.Text = "";
      this.txtNotes.Text = "";

      txtStartHuffman.Text = "";
      txtStartHuffmanSize.Text = "";
      txtPrecision.Text = "";
      txtNumberHuffmanLines.Text = "";
      txtNumberHuffmanSamples.Text = "";
      txtNumberImageComponents.Text = "";
      txtComponents.Text = "";

      // Label fields to clear
      this.lblApplicationMarker1.Text = "";
      this.lblApplicationMarker2.Text = "";
      this.lblApplicationMarker3.Text = "";
      this.lblApplicationMarker4.Text = "";
      this.lblApplicationMarker5.Text = "";
      this.lblApplicationMarker6.Text = "";
      this.lblApplicationMarker7.Text = "";
      this.lblApplicationMarker8.Text = "";
      this.lblApplicationMarker9.Text = "";
      this.lblApplicationMarker10.Text = "";

      this.lblExpandMarker.Text = "";
      this.lblHierarchialMarker.Text = "";
      this.lblNumberLinesMarker.Text = "";
      this.lblRestartMarker.Text = "";

      this.lblHuffmanMarker1.Text = "";
      this.lblHuffmanMarker2.Text = "";
      this.lblHuffmanMarker3.Text = "";
      this.lblHuffmanMarker4.Text = "";
      this.lblHuffmanMarker5.Text = "";
      this.lblHuffmanMarker6.Text = "";
      this.lblHuffmanMarker7.Text = "";
      this.lblHuffmanMarker8.Text = "";

      this.lblHuffmanOriginalMarker1.Text = "";
      this.lblHuffmanOriginalMarker2.Text = "";
```



```
      this.lblHuffmanOriginalMarker3.Text = "";
      this.lblHuffmanOriginalMarker4.Text = "";
      this.lblHuffmanOriginalMarker5.Text = "";
      this.lblHuffmanOriginalMarker6.Text = "";
      this.lblHuffmanOriginalMarker7.Text = "";
      this.lblHuffmanOriginalMarker8.Text = "";

      this.lblQuantizerMarker1.Text = "";
      this.lblQuantizerMarker2.Text = "";
      this.lblQuantizerMarker3.Text = "";
      this.lblQuantizerMarker4.Text = "";

      this.lblQuantizerOriginalMarker1.Text = "";
      this.lblQuantizerOriginalMarker2.Text = "";
      this.lblQuantizerOriginalMarker3.Text = "";
      this.lblQuantizerOriginalMarker4.Text = "";

      // Picture components to clear
      picOriginal.Image = null;
      picOriginal.Update();
      picOriginalSmall.Image = null;
      picOriginalSmall.Update();
      picManipulated.Image = null;
      picManipulated.Update();
      picManipulatedSmall.Image = null;
      picManipulatedSmall.Update();
    }


    /// <summary>
    /// Pre-conditions:    None.
    /// Post-conditions:
    ///    A new file with the data contained in the ByteDataToWrite array
    ///    has been created.
    ///  Description:
    ///    The Purpose of this function is to allow the caller to create a
    ///    new file based upon the data in the byte array passed in.  This
    ///    file created should be the binary value of the byte array and
    ///    nothing more.  If the byte array is null then an empty file
    ///    should be created.  The name of this file will be based upon file
    ///    name in the txtManipulatedFile TextBox control.  Lastly, this
    ///    method should do some error checking to make sure this function
    ///    executes properly.  If an error is encountered, then the
    ///    ShowWarning() method should be called to display the error to the
    ///    user and the txtError TextBox control should be updated with this
    ///    error information.
    /// </summary>
    /// <param name="ByteDataToWrite">The ByteDataToWrite parameter is byte
    /// array of data to be written to file.</param>
    private void WriteFile(ref byte[] ByteDataToWrite)
    {
      try
      {
        int c = FileSize;

        // Open the Original File to Setup Data
        if(NewFile != null) NewFile.Close();
        if(File.Exists(txtManipulatedFile.Text))
                        File.Delete(txtManipulatedFile.Text);
        NewFile = File.OpenWrite(this.txtManipulatedFile.Text);

        if (c >= ByteDataToWrite.Length) c = ByteDataToWrite.Length;
        NewFile.Write(ByteDataToWrite, 0, c);

        // Close the file when complete
        NewFile.Close();
      }
      catch(Exception EX)
      {
```

```
      // Catch some exceptions
      if(!ShowWarning(
        "An EXCEPTION occured!!  Exception: \n" +
        EX.Message + "\n\nThrown by: \n" + EX.Source +
        "\n\nWould you like to TRY to continue? \n" +
        "(If you choose OK, unexpected results may occur!)",
        "An Exception Occured!"))
      {
        ClearInterfaceData();
      }
    }
  }


  /// <summary>
  /// Pre-conditions:    None.
  /// Post-conditions:
  ///    All of the data members used to store information about the file
  ///    structure of the current JPEG image are reinitialized to zero.
  ///  Description:
  ///    The purpose of this method is to allow the caller to reinitialize
  ///    all of the data members that store information about the structure
  ///    of the previous JPEG image loaded.  This function should set the
  ///    following data members to zero: NumberOfLines, RestartInterval,
  ///    FrameSize, ExpandImage, RestartMod8, SizeOfHuffman (all 8 array
  ///    members), SizeOfQuantizer (all 4 array members), SizeOfAppData
  ///    (all 10 array members), SizeOfScanHeader, SizeOfProgression and
  ///    SizeOfComments.  Also, the FileOrder Queue should be cleared.
  /// </summary>
  private void ClearData()
  {
    int i = 0;

    NumberOfLines = 0;
    RestartInterval = 0;
    FrameSize = 0;
    ExpandImage = 0;
    RestartMod8 = 0;


    FileOrder.Clear();

    for(i = 0; i < MAX_HUFFMAN; i++) SizeOfHuffman[i] = 0;
    for(i = 0; i < MAX_QUANTIZER; i++) SizeOfQuantizer[i] = 0;
    for(i = 0; i < MAX_APPDATA; i++) SizeOfAppData[i] = 0;

    SizeOfScanHeader = 0;
    SizeOfProgression = 0;
    SizeOfComments = 0;

  }


  /// <summary>
  /// Pre-conditions:    None.
  /// Post-conditions:
  ///    A previously existing SEP project file has been reloaded into the
  ///    Manipulator.
  ///  Description:
  ///    The purpose of the function is to allow the caller to load a
  ///    pre-existing SEP project file.  This function should prompt the
  ///    user to save the current project, if there is one currently
  ///    loaded.  Then this function should call the ClearInterfaceData()
  ///    method and then should open the file and read all data, to reload
  ///    all of the corresponding fields in the interface.  This method
  ///    should load the project notes stored in the SEP file into the
  ///    txtNotes TextBox interface control.  This method should also
  ///    reload all of the PictureBox controls from the image file
  ///    information stored in the SEP file.  This method should do some
```



```
  ///    error checking to make sure all of the images load and that this
  ///    method executes properly.  If there is an error, the
  ///    ShowWarning() method should be called and the txtError TextBox
  ///    control should be updated with this error information.
  /// </summary>
  private void LoadNewProject()
  {
    this.tabProject.Focus();
    this.Update();

    try
    {
      openFileDialog1.ShowHelp = false;
      if(openFileDialog1.ShowDialog() != DialogResult.OK) return;

      if(txtProjectPath.Text != "")
      {
        if(!ShowWarning(
          "\nYou currently have a file open for editing.\n" +
          "If you open a newfile, all unsaved data will be lost!\n" +
          "Are you sure you want to open this new file?"))
        {
          return;
        }
      } // End of: if(txtProjectPath.Text != "")

      if(txtProjectPath.Text.Trim() != "")
      {
        if(!ShowWarning(
          "\nYou currently have a file open for editing.\n" +
          "If you open a newfile, all unsaved data will be lost!\n" +
          "Are you sure you want to open this new file?"))
        {
          return;
        }
      } // End of: if(txtProjectPath.Text != "")

      else if(txtOriginalFile.Text.Trim() != "")
      {
        if(!ShowWarning(
          "\nYou currently have a picture file open for editing.\n" +
          "If you open a newfile, all unsaved data will be lost!\n" +
          "Are you sure you want to open this new file?"))
        {
          return;
        }
      }

      // Clear the interface
      ClearInterfaceData();
      txtProjectPath.Text = openFileDialog1.FileName;

      // Open the file to read from
      StreamReader sr = new StreamReader(openFileDialog1.FileName);

      string S, original_file_path, changed_file_path;
      char [] Data = null;
      int Size;

      //
      // Read the data from SEP file
      //

      original_file_path = "";
      changed_file_path = "";

      // Get the Notes data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
```

```
        if(Size > 0)
        {
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtNotes.Text += Data[i].ToString();
          Data = null;
        }

        //
        // File Tab Data
        //

        // Get the Original File Path
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            original_file_path += Data[i].ToString();
          Data = null;
        }

        // Get the Manipulated File Path
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            changed_file_path += Data[i].ToString();
          Data = null;
        }


        if(File.Exists(original_file_path))
        {
          LoadPicture(original_file_path, changed_file_path);
        }
        else
        {
          if(ShowWarning(
            "The Original Picture file path:\n" + original_file_path +
            "\n\nsaved in this project is NO LONGER VALID!!" +
            "\n\nDo you want to browse to the picture location?",
            "Invalid File Path!!"))
          {
            LoadNewPicture();
          }
          else
          {
            ShowWarning("Load Project operation has been canceled.",
              "Load Project Canceled");
            ClearInterfaceData();
            return;
          }
        }

        // Get the File Size data
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtFileSize.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
```



```
          for(int i = 0; i < Data.Length; i++)
            txtFileSize.Text += Data[i].ToString();
          Data = null;
        }

        // Get the File Comments
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtComments.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtComments.Text += Data[i].ToString();
          Data = null;
        }

        //
        // Header Tab Data
        //

        // Get the Start of Compression Marker
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtStartHuffman.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtStartHuffman.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Start of Compression Header Size
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
                    txtStartHuffmanSize.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtStartHuffmanSize.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Precision
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtPrecision.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtPrecision.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Huffman Lines
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtNumberHuffmanLines.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
```

```
        for(int i = 0; i < Data.Length; i++)
          txtNumberHuffmanLines.Text += Data[i].ToString();
        Data = null;
      }

      // Get the Huffman Samples
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtNumberHuffmanSamples.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtNumberHuffmanSamples.Text += Data[i].ToString();
        Data = null;
      }

      // Get the Number of Image Components
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtNumberImageComponents.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtNumberImageComponents.Text += Data[i].ToString();
        Data = null;
      }

      // Get the Number of Components
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtComponents.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtComponents.Text += Data[i].ToString();
        Data = null;
      }

      //
      // Huffman Table Data
      //

      // Get Compression Table 1 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffman1.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffman1.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffman1.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
```



```
          txtHuffman1.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffmanOriginal1.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffmanOriginal1.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffmanOriginal1.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffmanOriginal1.Text += Data[i].ToString();
        Data = null;
      }

      // Get Compression Table 2 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffman2.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffman2.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffman2.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffman2.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffmanOriginal2.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffmanOriginal2.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
```

```
        txtHuffmanOriginal2.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffmanOriginal2.Text += Data[i].ToString();
        Data = null;
      }

      // Get Compression Table 3 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffman3.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffman3.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffman3.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffman3.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffmanOriginal3.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffmanOriginal3.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffmanOriginal3.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffmanOriginal3.Text += Data[i].ToString();
        Data = null;
      }

      // Get Compression Table 4 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffman4.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffman4.Text += Data[i].ToString();
        Data = null;
      }
```



```
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffman4.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffman4.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffmanOriginal4.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffmanOriginal4.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffmanOriginal4.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffmanOriginal4.Text += Data[i].ToString();
        Data = null;
      }

      // Get Compression Table 5 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffman5.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffman5.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffman5.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffman5.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffmanOriginal5.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
```

```
        for(int i = 0; i < Data.Length; i++)
          lblHuffmanOriginal5.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffmanOriginal5.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffmanOriginal5.Text += Data[i].ToString();
        Data = null;
      }

      // Get Compression Table 6 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffman6.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffman6.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffman6.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffman6.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffmanOriginal6.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffmanOriginal6.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffmanOriginal6.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffmanOriginal6.Text += Data[i].ToString();
        Data = null;
      }

      // Get Compression Table 7 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
```



```
      if(Size > 0)
      {
        lblHuffman7.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffman7.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffman7.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffman7.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffmanOriginal7.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffmanOriginal7.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffmanOriginal7.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffmanOriginal7.Text += Data[i].ToString();
        Data = null;
      }

      // Get Compression Table 8 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHuffman8.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHuffman8.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHuffman8.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHuffman8.Text += Data[i].ToString();
        Data = null;
```

```
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblHuffmanOriginal8.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblHuffmanOriginal8.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtHuffmanOriginal8.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtHuffmanOriginal8.Text += Data[i].ToString();
          Data = null;
        }

        //
        // Quantizer Table Data
        //

        // Get the Quantizer Table 1 Data
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblQuantizerMarker1.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblQuantizerMarker1.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizerTableNum1.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizerTableNum1.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizer1.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizer1.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
```



```
        if(Size > 0)
        {
          lblQuantizerOriginalMarker1.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblQuantizerOriginalMarker1.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizerOriginal1.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizerOriginal1.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Quantizer Table 2 Data
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblQuantizerMarker2.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblQuantizerMarker2.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizerTableNum2.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizerTableNum2.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizer2.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizer2.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblQuantizerOriginalMarker2.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblQuantizerOriginalMarker2.Text += Data[i].ToString();
          Data = null;
```

```
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizerOriginal2.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizerOriginal2.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Quantizer Table 3 Data
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblQuantizerMarker3.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblQuantizerMarker3.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizerTableNum3.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizerTableNum3.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizer3.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizer3.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblQuantizerOriginalMarker3.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblQuantizerOriginalMarker3.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizerOriginal3.Text = "";
          Data = new char [Size];
```



```
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizerOriginal3.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Quantizer Table 4 Data
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblQuantizerMarker4.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblQuantizerMarker4.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizerTableNum4.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizerTableNum4.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizer4.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizer4.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblQuantizerOriginalMarker4.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblQuantizerOriginalMarker4.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtQuantizerOriginal4.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtQuantizerOriginal4.Text += Data[i].ToString();
          Data = null;
        }

        //
        // Application Data
```

```
        //

        // Get the Application Data 1
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker1.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker1.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData1.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData1.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 2
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker2.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker2.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData2.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData2.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 3
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker3.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker3.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
```



```
          txtApplicationData3.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData3.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 4
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker4.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker4.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData4.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData4.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 5
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker5.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker5.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData5.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData5.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 6
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker6.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker6.Text += Data[i].ToString();
          Data = null;
```

```
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData6.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData6.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 7
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker7.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker7.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData7.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData7.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 8
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker8.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker8.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData8.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData8.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 9
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
```



```
          lblApplicationMarker9.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker9.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData9.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData9.Text += Data[i].ToString();
          Data = null;
        }

        // Get the Application Data 10
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblApplicationMarker10.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblApplicationMarker10.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtApplicationData10.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            txtApplicationData10.Text += Data[i].ToString();
          Data = null;
        }

        //
        // Misc Tab Data
        //

        // Get the Restart Marker Data
        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          lblRestartMarker.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
          for(int i = 0; i < Data.Length; i++)
            lblRestartMarker.Text += Data[i].ToString();
          Data = null;
        }

        S = sr.ReadLine();
        Size = System.Convert.ToInt32(S.Trim());
        if(Size > 0)
        {
          txtRestart.Text = "";
          Data = new char [Size];
          sr.Read(Data, 0, Size);
```

```
        for(int i = 0; i < Data.Length; i++)
          txtRestart.Text += Data[i].ToString();
        Data = null;
      }

      // Get the Number of Lines Marker Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblNumberLinesMarker.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblNumberLinesMarker.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtNumberLines.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtNumberLines.Text += Data[i].ToString();
        Data = null;
      }

      // Get the Expand Marker Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblExpandMarker.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblExpandMarker.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtExpand.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtExpand.Text += Data[i].ToString();
        Data = null;
      }

      // Get the Restart Mod 8 Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtRestartMod8.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtRestartMod8.Text += Data[i].ToString();
        Data = null;
      }

      // Get the Hierarchical Data
```



```
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        lblHierarchialMarker.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          lblHierarchialMarker.Text += Data[i].ToString();
        Data = null;
      }

      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtHierarchial.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtHierarchial.Text += Data[i].ToString();
        Data = null;
      }

      // Get the Error Data
      S = sr.ReadLine();
      Size = System.Convert.ToInt32(S.Trim());
      if(Size > 0)
      {
        txtError.Text = "";
        Data = new char [Size];
        sr.Read(Data, 0, Size);
        for(int i = 0; i < Data.Length; i++)
          txtError.Text += Data[i].ToString();
        Data = null;
      }


      // Close the Stream Reader
      sr.Close();

    } // End of: try block
    catch(Exception ex)
    {
      if(ex.Message == "Invalid parameter used." ||
        ex.Message == "A generic error occurred in GDI+." ||
        ex.Source == "System.Drawing")
      {
        string x = ProgramDirectory + @"\default_bad.jpg";
        LoadPicture(x, x);
      }
      else
      {
        ShowWarning(
          "Warning, an exception occured:\n\n" +
          "Exception Error:\n" +
          ex.Message + "\n\nWas throw by:\n" +
          ex.Source +
          "\n\nNot all load operations completed.!",
          "Load File Exception");
        ClearInterfaceData();
      }
    }
  }


  /// <summary>
  /// Pre-conditions:    None.
  /// Post-conditions:
```

```
    ///    All of the current values loaded in the Manipulator, any project
    ///    notes and current image file names have been saved in a SEP
    ///    project file name based upon the file name string in the
    ///    txtProjectPath TextBox control.
    ///  Description:
    ///    The purpose of this method is to allow the caller to save an SEP
    ///    project file based upon the current values loaded in the
    ///    interface of the Manipulator.  The data saved should include both
    ///    the file name and paths of the images currently loaded within the
    ///    Manipulator and all of the data in the TextBox controls on the
    ///    sub-tabs located under the Console tab, including the txtNotes
    ///    control for the project notes. The project name should be the
    ///    file name and path stored in the txtProjectPath TextBox control.
    ///    If a file with this name already exists, the user should be asked
    ///    if it is okay to overwrite the pre-existing project file.  Lastly,
    ///    this method should do some error checking to make sure this
    ///    function executes properly.  If an error is encountered, then the
    ///    ShowWarning() method should be called to display the error to the
    ///    user and the txtError TextBox control should be updated with this
    ///    error information.
    /// </summary>
    private void SaveNewProject()
    {
      // Check to make sure a JPEG is loaded.
      if(txtOriginalFile.Text == "" || !File.Exists(txtOriginalFile.Text))
      {
        ShowWarning(
          "There is NO JPEG file currently loaded!\n" +
          "Project WILL NOT be saved!",
          "Save Project Canceled");
        return;
      }

      // Show the save dialog box
      saveFileDialog1.ShowHelp = false;
      if(saveFileDialog1.ShowDialog() != DialogResult.OK) return;

      // Show warning if file already exists
      // If the users chooses OK, we'll overwrite the file.
      while(File.Exists(saveFileDialog1.FileName.Trim()))
      {
        if(!ShowWarning(
          "Project ALREADY exists!!\n\n" + saveFileDialog1.FileName +
          "\n\nWould you like to overwrite this file?",
          "Project File Already Exists!"))
        {
          if(saveFileDialog1.ShowDialog() != DialogResult.OK) return;
        }
        else break;
      }
      txtProjectPath.Text = saveFileDialog1.FileName.Trim();
      if(File.Exists(txtProjectPath.Text)) File.Delete(txtProjectPath.Text);

      try
      {
        // Create a file to write to
        StreamWriter sr;
        int Size;
        StringBuilder ProjData = new
          StringBuilder(AVE_FILE_SIZE, MAX_FILE_SIZE);

        //
        // Get all the data from the Manipulator in a String for Conversion
        //

        // Write size of the Notes and then the Data
        Size = txtNotes.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtNotes.Text.TrimEnd());
```



```
        //
        // File Tab Data
        //

        // Write the Original Picture path
        Size = txtOriginalFile.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtOriginalFile.Text.TrimEnd());

        // Write the Manipulated Picture path
        if(File.Exists(txtManipulatedFile.Text.TrimEnd()))
        {
          Size = txtManipulatedFile.Text.TrimEnd().Length;
          ProjData.Append(Size.ToString() + "\n");
          if(Size > 0) ProjData.Append(txtManipulatedFile.Text.TrimEnd());
        }
        else
        {
          Size = txtOriginalFile.Text.TrimEnd().Length;
          ProjData.Append(Size.ToString() + "\n");
          if(Size > 0) ProjData.Append(txtOriginalFile.Text.TrimEnd());
        }

        // Write the File Size data
        Size = txtFileSize.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtFileSize.Text.TrimEnd());

        // Write the Comments
        Size = txtComments.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtComments.Text.TrimEnd());

        //
        // Header Tab Data
        //

        // Write the Start of Compression Marker
        Size = txtStartHuffman.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtStartHuffman.Text.TrimEnd());

        // Write the Start of Compression Header Size
        Size = txtStartHuffmanSize.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtStartHuffmanSize.Text.TrimEnd());

        // Write the Precision
        Size = txtPrecision.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtPrecision.Text.TrimEnd());

        // Write the Huffman Lines
        Size = txtNumberHuffmanLines.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtNumberHuffmanLines.Text.TrimEnd());

        // Write the Huffman Samples
        Size = txtNumberHuffmanSamples.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtNumberHuffmanSamples.Text.TrimEnd());

        // Write the Number of Image Components
        Size = txtNumberImageComponents.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtNumberImageComponents.Text.TrimEnd());

        // Write the Number of Components
```

```
        Size = txtComponents.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtComponents.Text.TrimEnd());

        //
        // Huffman Table Data
        //

        // Write Compression Table 1 Data
        Size = lblHuffman1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffman1.Text.TrimEnd());

        Size = txtHuffman1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffman1.Text.TrimEnd());

        Size = lblHuffmanOriginal1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffmanOriginal1.Text.TrimEnd());

        Size = txtHuffmanOriginal1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffmanOriginal1.Text.TrimEnd());

        // Write Compression Table 2 Data
        Size = lblHuffman2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffman2.Text.TrimEnd());

        Size = txtHuffman2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffman2.Text.TrimEnd());

        Size = lblHuffmanOriginal2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffmanOriginal2.Text.TrimEnd());

        Size = txtHuffmanOriginal2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffmanOriginal2.Text.TrimEnd());

        // Write Compression Table 3 Data
        Size = lblHuffman3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffman3.Text.TrimEnd());

        Size = txtHuffman3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffman3.Text.TrimEnd());

        Size = lblHuffmanOriginal3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffmanOriginal3.Text.TrimEnd());

        Size = txtHuffmanOriginal3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffmanOriginal3.Text.TrimEnd());

        // Write Compression Table 4 Data
        Size = lblHuffman4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffman4.Text.TrimEnd());

        Size = txtHuffman4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffman4.Text.TrimEnd());

        Size = lblHuffmanOriginal4.Text.TrimEnd().Length;
```



```
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffmanOriginal4.Text.TrimEnd());

        Size = txtHuffmanOriginal4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffmanOriginal4.Text.TrimEnd());

        // Write Compression Table 5 Data
        Size = lblHuffman5.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffman5.Text.TrimEnd());

        Size = txtHuffman5.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffman5.Text.TrimEnd());

        Size = lblHuffmanOriginal5.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffmanOriginal5.Text.TrimEnd());

        Size = txtHuffmanOriginal5.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffmanOriginal5.Text.TrimEnd());

        // Write Compression Table 6 Data
        Size = lblHuffman6.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffman6.Text.TrimEnd());

        Size = txtHuffman6.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffman6.Text.TrimEnd());

        Size = lblHuffmanOriginal6.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffmanOriginal6.Text.TrimEnd());

        Size = txtHuffmanOriginal6.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffmanOriginal6.Text.TrimEnd());

        // Write Compression Table 7 Data
        Size = lblHuffman7.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffman7.Text.TrimEnd());

        Size = txtHuffman7.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffman7.Text.TrimEnd());

        Size = lblHuffmanOriginal7.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffmanOriginal7.Text.TrimEnd());

        Size = txtHuffmanOriginal7.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffmanOriginal7.Text.TrimEnd());

        // Write Compression Table 8 Data
        Size = lblHuffman8.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHuffman8.Text.TrimEnd());

        Size = txtHuffman8.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffman8.Text.TrimEnd());

        Size = lblHuffmanOriginal8.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
```

```
        if(Size > 0) ProjData.Append(lblHuffmanOriginal8.Text.TrimEnd());

        Size = txtHuffmanOriginal8.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHuffmanOriginal8.Text.TrimEnd());

        //
        // Quantizer Table Data
        //

        // Write the Quantizer Table 1 Data
        Size = lblQuantizerMarker1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblQuantizerMarker1.Text.TrimEnd());

        Size = txtQuantizerTableNum1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizerTableNum1.Text.TrimEnd());

        Size = txtQuantizer1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizer1.Text.TrimEnd());

        Size = lblQuantizerOriginalMarker1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0)
                        ProjData.Append(lblQuantizerOriginalMarker1.Text.TrimEnd
());

        Size = txtQuantizerOriginal1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizerOriginal1.Text.TrimEnd());

        // Write the Quantizer Table 2 Data
        Size = lblQuantizerMarker2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblQuantizerMarker2.Text.TrimEnd());

        Size = txtQuantizerTableNum2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizerTableNum2.Text.TrimEnd());

        Size = txtQuantizer2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizer2.Text.TrimEnd());

        Size = lblQuantizerOriginalMarker2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0)
                        ProjData.Append(lblQuantizerOriginalMarker2.Text.TrimEnd
());

        Size = txtQuantizerOriginal2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizerOriginal2.Text.TrimEnd());

        // Write the Quantizer Table 3 Data
        Size = lblQuantizerMarker3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblQuantizerMarker3.Text.TrimEnd());

        Size = txtQuantizerTableNum3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizerTableNum3.Text.TrimEnd());

        Size = txtQuantizer3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizer3.Text.TrimEnd());
```



```
        Size = lblQuantizerOriginalMarker3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0)
                        ProjData.Append(lblQuantizerOriginalMarker3.Text.TrimEnd
());

        Size = txtQuantizerOriginal3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizerOriginal3.Text.TrimEnd());

        // Write the Quantizer Table 4 Data
        Size = lblQuantizerMarker4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblQuantizerMarker4.Text.TrimEnd());

        Size = txtQuantizerTableNum4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizerTableNum4.Text.TrimEnd());

        Size = txtQuantizer4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizer4.Text.TrimEnd());

        Size = lblQuantizerOriginalMarker4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0)
                        ProjData.Append(lblQuantizerOriginalMarker4.Text.TrimEnd
());

        Size = txtQuantizerOriginal4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtQuantizerOriginal4.Text.TrimEnd());

        //
        // Application Data
        //

        // Write the Application Data 1
        Size = lblApplicationMarker1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker1.Text.TrimEnd());

        Size = txtApplicationData1.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData1.Text.TrimEnd());

        // Write the Application Data 2
        Size = lblApplicationMarker2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker2.Text.TrimEnd());

        Size = txtApplicationData2.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData2.Text.TrimEnd());

        // Write the Application Data 3
        Size = lblApplicationMarker3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker3.Text.TrimEnd());

        Size = txtApplicationData3.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData3.Text.TrimEnd());

        // Write the Application Data 4
        Size = lblApplicationMarker4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker4.Text.TrimEnd());
```

```
        Size = txtApplicationData4.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData4.Text.TrimEnd());

        // Write the Application Data 5
        Size = lblApplicationMarker5.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker5.Text.TrimEnd());

        Size = txtApplicationData5.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData5.Text.TrimEnd());

        // Write the Application Data 6
        Size = lblApplicationMarker6.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker6.Text.TrimEnd());

        Size = txtApplicationData6.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData6.Text.TrimEnd());

        // Write the Application Data 7
        Size = lblApplicationMarker7.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker7.Text.TrimEnd());

        Size = txtApplicationData7.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData7.Text.TrimEnd());

        // Write the Application Data 8
        Size = lblApplicationMarker8.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker8.Text.TrimEnd());

        Size = txtApplicationData8.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData8.Text.TrimEnd());

        // Write the Application Data 9
        Size = lblApplicationMarker9.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker9.Text.TrimEnd());

        Size = txtApplicationData9.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData9.Text.TrimEnd());

        // Write the Application Data 10
        Size = lblApplicationMarker10.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblApplicationMarker10.Text.TrimEnd());

        Size = txtApplicationData10.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtApplicationData10.Text.TrimEnd());

        //
        // Misc Tab Data
        //

        // Write the Restart Marker Data
        Size = lblRestartMarker.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblRestartMarker.Text.TrimEnd());

        Size = txtRestart.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
```



```
        if(Size > 0) ProjData.Append(txtRestart.Text.TrimEnd());

        // Write the Number of Lines Marker Data
        Size = lblNumberLinesMarker.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblNumberLinesMarker.Text.TrimEnd());

        Size = txtNumberLines.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtNumberLines.Text.TrimEnd());

        // Write the Expand Marker Data
        Size = lblExpandMarker.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblExpandMarker.Text.TrimEnd());

        Size = txtExpand.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtExpand.Text.TrimEnd());

        // Write the Restart Mod 8 Data
        Size = txtRestartMod8.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtRestartMod8.Text.TrimEnd());

        // Write the Hierarchical Data
        Size = lblHierarchialMarker.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(lblHierarchialMarker.Text.TrimEnd());

        Size = txtHierarchial.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtHierarchial.Text.TrimEnd());

        // Write the Error Data
        Size = txtError.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtError.Text.TrimEnd());

        //
        // Encoded Data Tab
        //

        // Write the Scan Header Data
        Size = txtScanHeader.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtScanHeader.Text.TrimEnd());

        Size = txtEncodedData.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtEncodedData.Text.TrimEnd());

        Size = txtOriginalHeader.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtOriginalHeader.Text.TrimEnd());

        Size = txtOriginalEncodedData.Text.TrimEnd().Length;
        ProjData.Append(Size.ToString() + "\n");
        if(Size > 0) ProjData.Append(txtOriginalEncodedData.Text.TrimEnd());

        //
        // Write the data to a file
        //
        sr = new StreamWriter(txtProjectPath.Text.Trim(), false);
        sr.Write(ProjData);
        sr.Close();
        sr = null;
      }
      catch(Exception EX)
```

```
    {
      ShowWarning(
        "Warning, an exception occured:\n\n" +
        "Exception Error:\n" +
        EX.Message + "\n\nWas throw by:\n" +
        EX.Source +
        "\n\nNot all save operations completed.!",
        "Save File Exception");
    }

  }


  #endregion Common Methods


  #region Methods to Convert from Binary to ACSII


  /// <summary>
  /// Pre-conditions:    None.
  /// Post-conditions:
  ///    The LowBits parameter is set to an ASCII character between 0 to F,
  ///    based upon the value of bits at positions 0 through 3 of the
  ///    bit-index of the OneByte parameter passed in.  The HighBits
  ///    parameter is set to an ASCII character of 0 to F, based upon the
  ///    value of bits at positions 4 through 7 of the bit-index of the
  ///    OneByte parameter passed in.
  ///  Description:
  ///    The purpose of this method is to allow the caller to easily
  ///    convert an 8-bit binary value to two ASCII characters representing
  ///    the hexadecimal value of these 8-bits.  To perform this
  ///    functionality, this method should split the OneByte parameter into
  ///    integer values, each with 4 bits in them.  Then, this function
  ///    should call the Convert() method that takes an integer and
  ///    returns a char for each of these two 4-bit values to get the
  ///    hexadecimal representation of each.  Then, each char should be
  ///    returned in the two reference parameters.
  /// </summary>
  /// <param name="OneByte">The OneByte parameter is an integer value
  /// between 0 and 255 (8-bits), representing the value of one
  /// byte.</param>
  /// <param name="HighBits">The HighBits parameter is a reference to a
  /// char where the char value resulting from the 4 most significant bits
  /// of the OneByte parameter can be stored.</param>
  /// <param name="LowBits">The LowBits parameter is a reference to a char
  /// where the char value resulting from the 4 least significant bits of
  /// the OneByte parameter can be stored.</param>
  private void SetCharValues(int OneByte, ref char HighBits,
              ref char LowBits)
  {
    High = OneByte % 256;  // Get 8 bits
    Low = High % 16;    // Get the bottom 4 bits
    High = High >> 4;    // Keep the top 4 bits
    HighBits = Convert(High);
    LowBits = (Convert(Low));
  }


  /// <summary>
  /// Pre-conditions:    None.
  ///  Post-conditions:
  ///    A character based on the hexadecimal value of the integer
  ///    parameter passed in should be returned.
  ///  Description:
  ///    The purpose of this function allows the caller to convert the
  ///    4-bit value of the parameter to an ASCII character representing
  ///    its hexadecimal value.  This function will return the character
  ///    M-^QXM-^R if the value of the parameter is not between the value of 0
```



```
  ///    and 15 and an error message box, txtError, will be displayed to
  ///    the user.
  /// </summary>
  /// <param name="Value">The Value parameter is an integer value between
  /// 0 and 15 (4-bits).</param>
  /// <returns>Function returns a char based upon the hexadecimal value of
  /// the parameter.</returns>
  private char Convert(int Value)
  {
    switch(Value)
    {
      case 0: return '0';
      case 1: return '1';
      case 2: return '2';
      case 3: return '3';
      case 4: return '4';
      case 5: return '5';
      case 6: return '6';
      case 7: return '7';
      case 8: return '8';
      case 9: return '9';
      case 10: return 'a';
      case 11: return 'b';
      case 12: return 'c';
      case 13: return 'd';
      case 14: return 'e';
      case 15: return 'f';
      default:
      {
        ShowWarning(
          "Function \"char Convert(int);\" encountered an unrecognized " +
          "character!!\nThis is a SERIOUS error! Please inform developer.");
        return 'X';
      }
    }
  }


  /// <summary>
  /// Pre-conditions:    None.
  /// Post-conditions:
  ///    All of the data for the JPEG image based upon the FilePath
  ///    parameter is loaded into all of the appropriate interface TextBox
  ///    controls for the user to view.
  ///  Description:
  ///    The purpose of this method is to load the binary file data for a
  ///    JPEG image into the all of the appropriate TextBox data fields
  ///    within the Manipulator interface.  This function opens the JPEG
  ///    file in binary mode and reads all the data from it.  Every byte
  ///    read from the file is converted to its hexadecimal representation
  ///    and is stored in the OriginalDataStream data member.  Then, to
  ///    load all of the data in the OriginalDataStream string in to the
  ///    interface, the LoadInterfaceData() method is called.  Lastly,
  ///    this method should do some error checking to make sure this
  ///    function executes properly.  If an error is encountered, then the
  ///    ShowWarning() method should be called to display the error to the
  ///    user and the txtError TextBox control should be updated with this
  ///    error information.
  /// </summary>
  /// <param name="FilePath">The FilePath parameter is the file name and
  /// path to a JPEG image.</param>
  private void LoadPictureData(string FilePath)
  {
    try
    {
      char Top1 = 'X';
      char Bottom1 = 'X';
```

```
      // Open the Original File to Setup Data
      if(OriginalFile != null) OriginalFile.Close();
      OriginalFile = File.OpenRead(FilePath);

      // Set start values
      OriginalDataStream.Length = 0;
      Value = 0;
      FileSize = 0;

      // Read out the file
      while(Value != -1)
      {
        Value = OriginalFile.ReadByte();
        if(Value == -1) break;
        FileSize++;
        SetCharValues(Value, ref Top1, ref Bottom1);
        OriginalDataStream.Append(Top1.ToString());
        OriginalDataStream.Append(Bottom1.ToString());
      }

      // Close the file when complete
      OriginalFile.Close();

      // Process the file string and load windows forms with data
      txtFileSize.Text = FileSize + " bytes";
      LoadInterfaceData(ref OriginalDataStream);

    }
    catch(Exception ex)
    {
      if(ShowWarning(
        "This program has encountered an UNHANDLED Exception!!\n\n" +
        ex.ToString() + "\n\nDo you want to close this program?",
        "Unhandled Exception Occurred!!"
        ))
      {
        menuExit.PerformClick();
      }

    }

  }


  /// <summary>
  /// Pre-conditions:    None.
  ///  Post-conditions:
  ///    All of the character data contained in the HexChars parameter is
  ///    broken apart and stored in the appropriate TextBox data fields in
  ///    the Manipulator.
  ///  Description:
  ///    The purpose of this method is to take an string of ASCII
  ///    characters that represent a JPEG file, break the file down into
  ///    its various frames and then input all of this data to its
  ///    corresponding TextBox data field in the interface.  As such, this
  ///    function is one of the largest functions in the Manipulator and
  ///    performs many tasks during its execution.  This method should read
  ///    through the data in the HexChars parameter passed in.  Every time
  ///    a file marker is found, it should be enqueued into the FileOrder
  ///    Queue data member.  Then, the data found behind this particular
  ///    marker should be loaded into its corresponding data field TextBox
  ///    control in the interface of the Manipulator.  Since we have to
  ///    account for every possible marker found within the JPEG standard,
  ///    this function should be implemented with a number of switch
  ///    statements to satisfy all possibilities. Also, as this function
  ///    encounters the different frames within the file, all of the
  ///    appropriate file structure data members of the JPEG Manipulator
  ///    should be set.  Lastly, this method should do lots of error
  ///    checking to make sure this function executes properly.  Items
```



```
  ///    to check for errors are possible errors in the structure or format
  ///    of the file and to make sure no exceptions occur when loading the
  ///    interface.  If an error is encountered, then the ShowWarning()
  ///    method should be called to display the error to the user and the
  ///    txtError TextBox control should be updated with this error
  ///    information.
  /// </summary>
  /// <param name="HexChars">The HexChars parameter contains the file data
  /// for a JPEG image converted to ASCII characters representing the
  /// hexadecimal value of each byte found in the original JPEG
  /// file.</param>
  private void LoadInterfaceData(ref StringBuilder HexChars)
  {
    char Top1 = 'X';
    char Bottom1 = 'X';

    bool Read = true;

    int FileLeng = HexChars.Length;
    int Count = 0;
    int Temp;
    Loading = new frmLoad();

    ClearData();

    EncodedData.Length = 0;
    Loading.StartLoading(0, FileLeng, 2);

    while(Count < FileLeng)
    {
      Top1 = HexChars[Count];
      Count++;
      Bottom1 = HexChars[Count];
      Count++;
      FileOrder.Enqueue(Top1);
      FileOrder.Enqueue(Bottom1);

      // Update the loading form
      Loading.UpdateAndIncrement();
      this.Update();

      if(Top1 == 'f' && Bottom1 == 'f')
      {

        // Read in the next byte to check file marker
        Top1 = HexChars[Count];
        Count++;
        Bottom1 = HexChars[Count];
        Count++;
        FileOrder.Enqueue(Top1);
        FileOrder.Enqueue(Bottom1);

        if(Top1 == 'd' && Bottom1 == '9') break;

        // Update the loading form and check for the Cancel button
        if(!Loading.UpdateAndIncrement())
        {
          if(ShowWarning(
            "You have choosen to cancel this load operation, " +
            "are you SURE you want to stop, " +
            "ALL loaded data will be LOST!\n\n " +
            "Are you sure you want to cancel?",
            "Cancel Loading?"))
          {
            ClearData();
            break;
          }
        }
```

```
        switch(Top1)
        {  // JPEG FILE MARKERS, Pg 106 in "JPEG" by: Pennebaker & Mitchell

          case '0': // Marker ff0X
          {
            switch(Bottom1)
            {
              case '0': // Marker ff00 - Marker Not Defined
              {
                txtError.Text +=
                  "\nError: Marker NOT defined " +
                  "\n\t-- Marker ff00 was found at byte index: " +
                  ((int)(Count - 4)).ToString();
                txtError.Update();
                break;
              }
              case '1': // Marker ff01
              {
                txtError.Text +=
                  "\nPossible Error: Marker found Temporary use for " +
                  "Arthmetic Encoding " +
                  "\n\t-- Marker ff01 was found at byte index: " +
                  ((int)(Count - 4)).ToString();
                txtError.Update();

                break;
              }
              case '2': goto case 'f';
              case '3': goto case 'f';
              case '4': goto case 'f';
              case '5': goto case 'f';
              case '6': goto case 'f';
              case '7': goto case 'f';
              case '8': goto case 'f';
              case '9': goto case 'f';
              case 'a': goto case 'f';
              case 'b': goto case 'f';
              case 'c': goto case 'f';
              case 'd': goto case 'f';
              case 'e': goto case 'f';
              case 'f':
              {
                // Marker ff02 to ff0f - Reserved
                txtError.Text +=
                  "\nPossible Error: Reserved Marker Found!! " +
                  "\n\t-- Marker ff0" + Bottom1.ToString()+
                  " was found at byte index: " +
                  ((int)(Count - 4)).ToString();
                txtError.Update();

                break;

              }

              default:
              {
                txtError.Text +=
                  "\nError: Invalid File Marker Read!! " +
                  "\n\t-- Marker ff0" + Bottom1.ToString()+
                  " was found at byte index: " +
                  ((int)(Count - 4)).ToString();
                txtError.Update();

                break;
              }

            } // End of: switch(Bottom1)

            break;
```



```
          } // End of: case '0';

          case '1': goto case 'b';
          case '2': goto case 'b';
          case '3': goto case 'b';
          case '4': goto case 'b';
          case '5': goto case 'b';
          case '6': goto case 'b';
          case '7': goto case 'b';
          case '8': goto case 'b';
          case '9': goto case 'b';
          case 'a': goto case 'b';
          case 'b':
          { // Marker ff10 to ffbf - Reserved
            txtError.Text +=
              "\nPossible Error: Reserved Marker Found!! " +
              "\n\t-- Marker ff" + Top1.ToString() + Bottom1.ToString()+
              " was found at byte index: " +
              ((int)(Count - 4)).ToString();
            txtError.Update();
            break;
          }

          case 'c': // marker ffcX - huffman tables
          {
            switch(Bottom1)
            {
              // Start of: Nondifferential Huffman-Coding Frames
              case '0': // marker ffc0 - Baseline DCT
              {
                string info;
                Read = false;

                txtStartHuffman.Text = "ffc0";

                // Read in the Frame Size to set values
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize = SetByteValue(Top1, Bottom1);
                FrameSize = FrameSize << 8;
                                    // to get the rest of the counter
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize += SetByteValue(Top1, Bottom1);

                // Load the size on the interface
                txtStartHuffmanSize.Text = FrameSize.ToString();

                // Update the loading form
                Loading.LoadProgressValue += 2;
                Loading.UpdateAndIncrement();
                this.Update();

                // Get Precision - 1 byte
                txtPrecision.Text = HexChars[Count].ToString();
                Count++;
                txtPrecision.Text += HexChars[Count].ToString();
                Count++;

                // Update the loading form
                Loading.UpdateAndIncrement();
                this.Update();

                // Get the number of lines - 2 bytes
```

```
                txtNumberHuffmanLines.Text = HexChars[Count].ToString();
                Count++;
                txtNumberHuffmanLines.Text += HexChars[Count].ToString();
                Count++;
                txtNumberHuffmanLines.Text += " ";
                txtNumberHuffmanLines.Text += HexChars[Count].ToString();
                Count++;
                txtNumberHuffmanLines.Text += HexChars[Count].ToString();
                Count++;

                // Update the loading form
                Loading.LoadProgressValue += 2;
                Loading.UpdateAndIncrement();
                this.Update();

                // Get the number of samples per line - 2 bytes
                txtNumberHuffmanSamples.Text = HexChars[Count].ToString();
                Count++;
                txtNumberHuffmanSamples.Text += HexChars[Count].ToString();
                Count++;
                txtNumberHuffmanSamples.Text += " ";
                txtNumberHuffmanSamples.Text += HexChars[Count].ToString();
                Count++;
                txtNumberHuffmanSamples.Text += HexChars[Count].ToString();
                Count++;

                // Update the loading form
                Loading.LoadProgressValue += 2;
                Loading.UpdateAndIncrement();
                this.Update();

                // Get number of image components - 1 byte
                txtNumberImageComponents.Text = HexChars[Count].ToString();
                Count++;
                txtNumberImageComponents.Text += HexChars[Count].ToString();
                Count++;
                FrameSize =
                                    SetByteValue(HexChars[Count-2], HexCha
rs[Count-1]);

                // Update the loading form
                Loading.UpdateAndIncrement();
                this.Update();

                info = "Identifier, Horizontal, Vertical, Q-Table: \n";
                txtComponents.Text = info;

                for(int a = FrameSize; a > 0; a--)
                {
                  // Component identifier
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  info = Top1.ToString() + Bottom1.ToString() + ", ";

                  // Horizontal and Vertical Sampling factor
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  info += Top1.ToString() + ", " +
                                                  Bottom1.ToString() + ",
";

                  // Quantization Table Selector
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
```



```
                  Count++;
                  info += Top1.ToString() + Bottom1.ToString();
                  txtComponents.Text += info;
                  txtComponents.Text += "\n";
                }

                break;
              }
              case '1': // marker ffc1 - Extended Sequential DCT
              {
                if(lblHuffmanMarker1.Text == "")
                                    lblHuffmanMarker1.Text = "ffc1";
                else if(lblHuffmanMarker2.Text == "")
                                    lblHuffmanMarker2.Text = "ffc1";
                else if(lblHuffmanMarker3.Text == "")
                                    lblHuffmanMarker3.Text = "ffc1";
                else if(lblHuffmanMarker4.Text == "")
                                    lblHuffmanMarker4.Text = "ffc1";
                else if(lblHuffmanMarker5.Text == "")
                                    lblHuffmanMarker5.Text = "ffc1";
                else if(lblHuffmanMarker6.Text == "")
                                    lblHuffmanMarker6.Text = "ffc1";
                else if(lblHuffmanMarker7.Text == "")
                                    lblHuffmanMarker7.Text = "ffc1";
                else if(lblHuffmanMarker8.Text == "")
                                    lblHuffmanMarker8.Text = "ffc1";
                break;
              }
              case '2': // marker ffc2 - Progressive DCT
              {
                if(lblHuffmanMarker1.Text == "")
                                    lblHuffmanMarker1.Text = "ffc2";
                else if(lblHuffmanMarker2.Text == "")
                                    lblHuffmanMarker2.Text = "ffc2";
                else if(lblHuffmanMarker3.Text == "")
                                    lblHuffmanMarker3.Text = "ffc2";
                else if(lblHuffmanMarker4.Text == "")
                                    lblHuffmanMarker4.Text = "ffc2";
                else if(lblHuffmanMarker5.Text == "")
                                    lblHuffmanMarker5.Text = "ffc2";
                else if(lblHuffmanMarker6.Text == "")
                                    lblHuffmanMarker6.Text = "ffc2";
                else if(lblHuffmanMarker7.Text == "")
                                    lblHuffmanMarker7.Text = "ffc2";
                else if(lblHuffmanMarker8.Text == "")
                                    lblHuffmanMarker8.Text = "ffc2";
                break;
              }
              case '3': // marker ffc3 - Lossless (Sequential)
              {
                if(lblHuffmanMarker1.Text == "")
                                    lblHuffmanMarker1.Text = "ffc3";
                else if(lblHuffmanMarker2.Text == "")
                                    lblHuffmanMarker2.Text = "ffc3";
                else if(lblHuffmanMarker3.Text == "")
                                    lblHuffmanMarker3.Text = "ffc3";
                else if(lblHuffmanMarker4.Text == "")
                                    lblHuffmanMarker4.Text = "ffc3";
                else if(lblHuffmanMarker5.Text == "")
                                    lblHuffmanMarker5.Text = "ffc3";
                else if(lblHuffmanMarker6.Text == "")
                                    lblHuffmanMarker6.Text = "ffc3";
                else if(lblHuffmanMarker7.Text == "")
                                    lblHuffmanMarker7.Text = "ffc3";
                else if(lblHuffmanMarker8.Text == "")
                                    lblHuffmanMarker8.Text = "ffc3";
                break;
              }
                // End of: Nondifferential Huffman-Coding Frames
```

```
case '4': // marker ffc4 - Define Huffman Marker
{
  if(lblHuffmanMarker1.Text == "")
                          lblHuffmanMarker1.Text = "ffc4";
  else if(lblHuffmanMarker2.Text == "")
                          lblHuffmanMarker2.Text = "ffc4";
  else if(lblHuffmanMarker3.Text == "")
                          lblHuffmanMarker3.Text = "ffc4";
  else if(lblHuffmanMarker4.Text == "")
                          lblHuffmanMarker4.Text = "ffc4";
  else if(lblHuffmanMarker5.Text == "")
                          lblHuffmanMarker5.Text = "ffc4";
  else if(lblHuffmanMarker6.Text == "")
                          lblHuffmanMarker6.Text = "ffc4";
  else if(lblHuffmanMarker7.Text == "")
                          lblHuffmanMarker7.Text = "ffc4";
  else if(lblHuffmanMarker8.Text == "")
                          lblHuffmanMarker8.Text = "ffc4";
  break;
}


  // Start of: Differential Huffman-Coding Frames
case '5': // marker ffc5 - Differential Sequential DCT
{
  if(lblHuffmanMarker1.Text == "")
                          lblHuffmanMarker1.Text = "ffc5";
  else if(lblHuffmanMarker2.Text == "")
                          lblHuffmanMarker2.Text = "ffc5";
  else if(lblHuffmanMarker3.Text == "")
                          lblHuffmanMarker3.Text = "ffc5";
  else if(lblHuffmanMarker4.Text == "")
                          lblHuffmanMarker4.Text = "ffc5";
  else if(lblHuffmanMarker5.Text == "")
                          lblHuffmanMarker5.Text = "ffc5";
  else if(lblHuffmanMarker6.Text == "")
                          lblHuffmanMarker6.Text = "ffc5";
  else if(lblHuffmanMarker7.Text == "")
                          lblHuffmanMarker7.Text = "ffc5";
  else if(lblHuffmanMarker8.Text == "")
                          lblHuffmanMarker8.Text = "ffc5";
  break;
}
case '6': // marker ffc6 - Differential Progressive DCT
{
  if(lblHuffmanMarker1.Text == "")
                          lblHuffmanMarker1.Text = "ffc6";
  else if(lblHuffmanMarker2.Text == "")
                          lblHuffmanMarker2.Text = "ffc6";
  else if(lblHuffmanMarker3.Text == "")
                          lblHuffmanMarker3.Text = "ffc6";
  else if(lblHuffmanMarker4.Text == "")
                          lblHuffmanMarker4.Text = "ffc6";
  else if(lblHuffmanMarker5.Text == "")
                          lblHuffmanMarker5.Text = "ffc6";
  else if(lblHuffmanMarker6.Text == "")
                          lblHuffmanMarker6.Text = "ffc6";
  else if(lblHuffmanMarker7.Text == "")
                          lblHuffmanMarker7.Text = "ffc6";
  else if(lblHuffmanMarker8.Text == "")
                          lblHuffmanMarker8.Text = "ffc6";
  break;
}
case '7': // marker ffc7 - Differential Lossless
{
  if(lblHuffmanMarker1.Text == "")
                          lblHuffmanMarker1.Text = "ffc7";
```



```
  else if(lblHuffmanMarker2.Text == "")
                          lblHuffmanMarker2.Text = "ffc7";
  else if(lblHuffmanMarker3.Text == "")
                          lblHuffmanMarker3.Text = "ffc7";
  else if(lblHuffmanMarker4.Text == "")
                          lblHuffmanMarker4.Text = "ffc7";
  else if(lblHuffmanMarker5.Text == "")
                          lblHuffmanMarker5.Text = "ffc7";
  else if(lblHuffmanMarker6.Text == "")
                          lblHuffmanMarker6.Text = "ffc7";
  else if(lblHuffmanMarker7.Text == "")
                          lblHuffmanMarker7.Text = "ffc7";
  else if(lblHuffmanMarker8.Text == "")
                          lblHuffmanMarker8.Text = "ffc7";
  break;
}
  // End of: Differential Huffman-Coding Frames


case '8': // marker ffc8 - Reserved for JPEG Extensions
{
  txtError.Text +=
    "\nPossible Error: Reserved For JPEG Extensions Marker"+
                          "Found!!\n\t-- Marker FFCD was found at
byte index: " +
    ((int)(Count - 4)).ToString();
  txtError.Update();
  Read = false; // Skip reading values for this marker
  break;
}


  // Start of: Nondifferential Arithmetic-Coding Frames
case '9': // marker ffc9 - Exteneded Sequential DCT
{
  if(lblHuffmanMarker1.Text == "")
                          lblHuffmanMarker1.Text = "ffc9";
  else if(lblHuffmanMarker2.Text == "")
                          lblHuffmanMarker2.Text = "ffc9";
  else if(lblHuffmanMarker3.Text == "")
                          lblHuffmanMarker3.Text = "ffc9";
  else if(lblHuffmanMarker4.Text == "")
                          lblHuffmanMarker4.Text = "ffc9";
  else if(lblHuffmanMarker5.Text == "")
                          lblHuffmanMarker5.Text = "ffc9";
  else if(lblHuffmanMarker6.Text == "")
                          lblHuffmanMarker6.Text = "ffc9";
  else if(lblHuffmanMarker7.Text == "")
                          lblHuffmanMarker7.Text = "ffc9";
  else if(lblHuffmanMarker8.Text == "")
                          lblHuffmanMarker8.Text = "ffc9";
  break;
}
case 'a': // marker ffca - Progressive DCT
{
  if(lblHuffmanMarker1.Text == "")
                          lblHuffmanMarker1.Text = "ffca";
  else if(lblHuffmanMarker2.Text == "")
                          lblHuffmanMarker2.Text = "ffca";
  else if(lblHuffmanMarker3.Text == "")
                          lblHuffmanMarker3.Text = "ffca";
  else if(lblHuffmanMarker4.Text == "")
                          lblHuffmanMarker4.Text = "ffca";
  else if(lblHuffmanMarker5.Text == "")
                          lblHuffmanMarker5.Text = "ffca";
  else if(lblHuffmanMarker6.Text == "")
                          lblHuffmanMarker6.Text = "ffca";
  else if(lblHuffmanMarker7.Text == "")
                          lblHuffmanMarker7.Text = "ffca";
```

```
                else if(lblHuffmanMarker8.Text == "")
                                lblHuffmanMarker8.Text = "ffca";
                break;
              }
              case 'b': // marker ffcb - Lossless (Sequential)
              {
                if(lblHuffmanMarker1.Text == "")
                                lblHuffmanMarker1.Text = "ffcb";
                else if(lblHuffmanMarker2.Text == "")
                                lblHuffmanMarker2.Text = "ffcb";
                else if(lblHuffmanMarker3.Text == "")
                                lblHuffmanMarker3.Text = "ffcb";
                else if(lblHuffmanMarker4.Text == "")
                                lblHuffmanMarker4.Text = "ffcb";
                else if(lblHuffmanMarker5.Text == "")
                                lblHuffmanMarker5.Text = "ffcb";
                else if(lblHuffmanMarker6.Text == "")
                                lblHuffmanMarker6.Text = "ffcb";
                else if(lblHuffmanMarker7.Text == "")
                                lblHuffmanMarker7.Text = "ffcb";
                else if(lblHuffmanMarker8.Text == "")
                                lblHuffmanMarker8.Text = "ffcb";
                break;
              }
                // End of: Nondifferential Arithmetic-Coding Frames


              case 'c': // marker ffcc - Define Arithmetic Conditioning Tables
              {
                if(lblHuffmanMarker1.Text == "")
                                lblHuffmanMarker1.Text = "ffcc";
                else if(lblHuffmanMarker2.Text == "")
                                lblHuffmanMarker2.Text = "ffcc";
                else if(lblHuffmanMarker3.Text == "")
                                lblHuffmanMarker3.Text = "ffcc";
                else if(lblHuffmanMarker4.Text == "")
                                lblHuffmanMarker4.Text = "ffcc";
                else if(lblHuffmanMarker5.Text == "")
                                lblHuffmanMarker5.Text = "ffcc";
                else if(lblHuffmanMarker6.Text == "")
                                lblHuffmanMarker6.Text = "ffcc";
                else if(lblHuffmanMarker7.Text == "")
                                lblHuffmanMarker7.Text = "ffcc";
                else if(lblHuffmanMarker8.Text == "")
                                lblHuffmanMarker8.Text = "ffcc";
                break;
              }


                // Start of: Differential Arithmetic-Coding Frames
              case 'd': // marker ffcd - Differential Sequential DCT
              {
                if(lblHuffmanMarker1.Text == "")
                                lblHuffmanMarker1.Text = "ffcd";
                else if(lblHuffmanMarker2.Text == "")
                                lblHuffmanMarker2.Text = "ffcd";
                else if(lblHuffmanMarker3.Text == "")
                                lblHuffmanMarker3.Text = "ffcd";
                else if(lblHuffmanMarker4.Text == "")
                                lblHuffmanMarker4.Text = "ffcd";
                else if(lblHuffmanMarker5.Text == "")
                                lblHuffmanMarker5.Text = "ffcd";
                else if(lblHuffmanMarker6.Text == "")
                                lblHuffmanMarker6.Text = "ffcd";
                else if(lblHuffmanMarker7.Text == "")
                                lblHuffmanMarker7.Text = "ffcd";
                else if(lblHuffmanMarker8.Text == "")
                                lblHuffmanMarker8.Text = "ffcd";
                break;
```



```
              }
              case 'e': // marker ffce - Differential Progressive DCT
              {
                if(lblHuffmanMarker1.Text == "")
                                lblHuffmanMarker1.Text = "ffce";
                else if(lblHuffmanMarker2.Text == "")
                                lblHuffmanMarker2.Text = "ffce";
                else if(lblHuffmanMarker3.Text == "")
                                lblHuffmanMarker3.Text = "ffce";
                else if(lblHuffmanMarker4.Text == "")
                                lblHuffmanMarker4.Text = "ffce";
                else if(lblHuffmanMarker5.Text == "")
                                lblHuffmanMarker5.Text = "ffce";
                else if(lblHuffmanMarker6.Text == "")
                                lblHuffmanMarker6.Text = "ffce";
                else if(lblHuffmanMarker7.Text == "")
                                lblHuffmanMarker7.Text = "ffce";
                else if(lblHuffmanMarker8.Text == "")
                                lblHuffmanMarker8.Text = "ffce";
                break;
              }
              case 'f': // marker ffcf - Differential Lossless
              {
                if(lblHuffmanMarker1.Text == "")
                                lblHuffmanMarker1.Text = "ffcf";
                else if(lblHuffmanMarker2.Text == "")
                                lblHuffmanMarker2.Text = "ffcf";
                else if(lblHuffmanMarker3.Text == "")
                                lblHuffmanMarker3.Text = "ffcf";
                else if(lblHuffmanMarker4.Text == "")
                                lblHuffmanMarker4.Text = "ffcf";
                else if(lblHuffmanMarker5.Text == "")
                                lblHuffmanMarker5.Text = "ffcf";
                else if(lblHuffmanMarker6.Text == "")
                                lblHuffmanMarker6.Text = "ffcf";
                else if(lblHuffmanMarker7.Text == "")
                                lblHuffmanMarker7.Text = "ffcf";
                else if(lblHuffmanMarker8.Text == "")
                                lblHuffmanMarker8.Text = "ffcf";
                break;
              }
                // End of: Differential Arithmetic-Coding Frames

              default:
              {
                txtError.Text +=
                  "\nError: Invalid File Marker Read!! " +
                  "\n\t-- Marker ffc" + Bottom1.ToString()+
                  " was found at byte index: " +
                  ((int)(Count - 4)).ToString();
                txtError.Update();

                break;
              }

            }  // End of: switch(Bottom1)

            if(Read)
            {
              // Read in the Frame Size to set values
              Top1 = HexChars[Count];
              Count++;
              Bottom1 = HexChars[Count];
              Count++;
              FrameSize = SetByteValue(Top1, Bottom1);
              FrameSize = FrameSize << 8;
                                // to get the rest of the counter
              Top1 = HexChars[Count];
              Count++;
```

```
            Bottom1 = HexChars[Count];
            Count++;
            FrameSize += SetByteValue(Top1, Bottom1);
            FrameSize -= 2;  // For the 2 bytes that hold the frame size

            // Update the loading form
            Loading.LoadProgressValue += 2;
            Loading.UpdateAndIncrement();
            this.Update();

            if(txtHuffman1.Text == "")
            {
              SizeOfHuffman[0] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtHuffman1.Text += Top1.ToString() +
                                    Bottom1.ToString() + " ";
              }

              // Update the loading form
              Loading.LoadProgressValue += SizeOfHuffman[0] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtHuffman2.Text == "")
            {
              SizeOfHuffman[1] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtHuffman2.Text += Top1.ToString() +
                                    Bottom1.ToString() + " ";
              }

              // Update the loading form
              Loading.LoadProgressValue += SizeOfHuffman[1] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtHuffman3.Text == "")
            {
              SizeOfHuffman[2] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtHuffman3.Text += Top1.ToString() +
                                    Bottom1.ToString() + " ";
              }

              // Update the loading form
              Loading.LoadProgressValue += SizeOfHuffman[2] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
```



```
            }
            else if(txtHuffman4.Text == "")
            {
              SizeOfHuffman[3] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtHuffman4.Text += Top1.ToString() +
                                    Bottom1.ToString() + " ";
              }

              // Update the loading form
              Loading.LoadProgressValue += SizeOfHuffman[3] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtHuffman5.Text == "")
            {
              SizeOfHuffman[4] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtHuffman5.Text += Top1.ToString() +
                                    Bottom1.ToString() + " ";
              }

              // Update the loading form
              Loading.LoadProgressValue += SizeOfHuffman[4] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtHuffman6.Text == "")
            {
              SizeOfHuffman[5] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtHuffman6.Text += Top1.ToString() +
                                    Bottom1.ToString() + " ";
              }

              // Update the loading form
              Loading.LoadProgressValue += SizeOfHuffman[5] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtHuffman7.Text == "")
            {
              SizeOfHuffman[6] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
```

```
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtHuffman7.Text += Top1.ToString() +
                                    Bottom1.ToString() + " ";
              }

              // Update the loading form
              Loading.LoadProgressValue += SizeOfHuffman[6] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtHuffman8.Text == "")
            {
              SizeOfHuffman[7] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtHuffman8.Text += Top1.ToString() +
                                    Bottom1.ToString() + " ";
              }

              // Update the loading form
              Loading.LoadProgressValue += SizeOfHuffman[7] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else
            {
              // Show an error.

            }

          } // End of: if(Read);
          else
          {
            Read = true;
          }

          break;
        }  // End of: case 'c': // marker ffcX

        case 'd': // marker ffdX
        {
          switch(Bottom1)
          {
            case '0': goto case '7';
            case '1': goto case '7';
            case '2': goto case '7';
            case '3': goto case '7';
            case '4': goto case '7';
            case '5': goto case '7';
            case '6': goto case '7';
            case '7':
            {  // Marker ffd0 to ffd7
              txtRestartMod8.Text = ((int)(Count - 4)).ToString();
              break;
            }

            case '8':
            {  // Marker ffd8 : Start of Image
              break;
            }
```



```
            case '9':
            {  // Marker ffd9 : End of image
              // Covered by: case ffda
              break;
            }

            case 'a':
            {  // Marker ffda : Start of Scan

              int i = 0;

              Top1 = 'X';
              Bottom1 = 'X';
              FrameSize = 0;

              // Get Scan Header
              Top1 = HexChars[Count];
              Count++;
              Bottom1 = HexChars[Count];
              Count++;
              FrameSize = SetByteValue(Top1, Bottom1);
              FrameSize = FrameSize << 8;
              Top1 = HexChars[Count];
              Count++;
              Bottom1 = HexChars[Count];
              Count++;
              FrameSize += SetByteValue(Top1, Bottom1);
              SizeOfScanHeader = FrameSize;
              FrameSize -= 2;

              // Update the loading form
              Loading.LoadProgressValue += 2;
              Loading.UpdateAndIncrement();
              this.Update();

              for(i = 0; i < FrameSize; i++)
              {
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                txtScanHeader.Text +=
                  Top1.ToString() + Bottom1.ToString() + " ";
              }
              txtScanHeader.Update();

              // Update the loading form
              Loading.LoadProgressValue +=
                                    ((txtScanHeader.Text.Length * 2)/3) -
2;
              Loading.UpdateAndIncrement();
              this.Update();

              // Get the encoded data stream
              temp = HexChars.Length - (Count + 4);
              EncodedData.Insert(0,
                                    HexChars.ToString().Substring(Count, t
emp));

              // Update the loading form
              Loading.LoadProgressValue += EncodedData.Length - 2;
              Loading.UpdateAndIncrement();
              this.Update();

              OriginalEncodedData = EncodedData.ToString();

              int MaxDisplay = 10240; // 5k in file size
              if(EncodedData.Length < MaxDisplay)
              {
```

```
                    txtEncodedData.Text = EncodedData.ToString();
                  }
                  else
                  {
                                      txtEncodedData.Text =
                                              EncodedData.ToString().Substri
ng(0, MaxDisplay);
                  }

                  Count += temp;

                  txtEncodedData.Update();

                  Top1 = 'f';
                  Bottom1 = 'f';

                  break;
                }

                case 'b':
                {  // Marker ffdb : Define Quantization Table

                  if(lblQuantizerMarker1.Text == "")
                                      lblQuantizerMarker1.Text = "ffdb";
                  else if(lblQuantizerMarker2.Text == "")
                                      lblQuantizerMarker2.Text = "ffdb";
                  else if(lblQuantizerMarker3.Text == "")
                                      lblQuantizerMarker3.Text = "ffdb";
                  else if(lblQuantizerMarker4.Text == "")
                                      lblQuantizerMarker4.Text = "ffdb";

                  // Read in the Frame Size to set values
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize = SetByteValue(Top1, Bottom1);
                  FrameSize = FrameSize << 8;
                                      // to get the rest of the counter
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize += SetByteValue(Top1, Bottom1);
                  FrameSize -= 2;
                                      // For the 2 bytes that hold the frame s
ize

                  // Update the loading form
                  Loading.LoadProgressValue += 2;
                  Loading.UpdateAndIncrement();
                  this.Update();

                  if(txtQuantizer1.Text == "")
                  {
                    // Read in the table Number - 1 byte
                    Top1 = HexChars[Count];
                    Count++;
                    Bottom1 = HexChars[Count];
                    Count++;
                    FrameSize--;
                    txtQuantizerTableNum1.Text =
                      Top1.ToString() + Bottom1.ToString();

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // 2 for framesize field and 1 for table number
```



```
                    SizeOfQuantizer[0] = FrameSize + 3;

                    // Read out the table data
                    while(FrameSize > 0)
                    {
                      // We are counting down the FrameSize to start the
                                          // stream.
                      Top1 = HexChars[Count];
                      Count++;
                      Bottom1 = HexChars[Count];
                      Count++;
                      FrameSize--;
                      txtQuantizer1.Text += Top1.ToString() +
                                                Bottom1.ToString() + " ";
                    }
                    // Update the loading form
                    Loading.LoadProgressValue += SizeOfQuantizer[0] - 2;
                    Loading.UpdateAndIncrement();
                    this.Update();

                  }
                  else if(txtQuantizer2.Text == "")
                  {
                    // Read in the table Number - 1 byte
                    Top1 = HexChars[Count];
                    Count++;
                    Bottom1 = HexChars[Count];
                    Count++;
                    FrameSize--;
                    txtQuantizerTableNum2.Text =
                      Top1.ToString() + Bottom1.ToString();

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // 2 for framesize field and 1 for table number
                    SizeOfQuantizer[1] = FrameSize + 3;

                    // Read out the table data
                    while(FrameSize > 0)
                    {
                      // We are counting down the FrameSize to start the
                                          // stream.
                      Top1 = HexChars[Count];
                      Count++;
                      Bottom1 = HexChars[Count];
                      Count++;
                      FrameSize--;
                      txtQuantizer2.Text += Top1.ToString() +
                                                Bottom1.ToString() + " ";
                    }
                    // Update the loading form
                    Loading.LoadProgressValue += SizeOfQuantizer[1] - 2;
                    Loading.UpdateAndIncrement();
                    this.Update();
                  }
                  else if(txtQuantizer3.Text == "")
                  {
                    // Read in the table Number - 1 byte
                    Top1 = HexChars[Count];
                    Count++;
                    Bottom1 = HexChars[Count];
                    Count++;
                    FrameSize--;
                    txtQuantizerTableNum3.Text =
                      Top1.ToString() + Bottom1.ToString();

                    // Update the loading form
```

```
                Loading.UpdateAndIncrement();
                this.Update();

                // 2 for framesize field and 1 for table number
                SizeOfQuantizer[2] = FrameSize + 3;

                // Read out the table data
                while(FrameSize > 0)
                {
                  // We are counting down the FrameSize to start the
                                  // stream.
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize--;
                  txtQuantizer3.Text += Top1.ToString() +
                                          Bottom1.ToString() + " ";
                }
                // Update the loading form
                Loading.LoadProgressValue += SizeOfQuantizer[2] - 2;
                Loading.UpdateAndIncrement();
                this.Update();
              }
              else if(txtQuantizer4.Text == "")
              {
                // Read in the table Number - 1 byte
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtQuantizerTableNum4.Text =
                  Top1.ToString() + Bottom1.ToString();

                // Update the loading form
                Loading.UpdateAndIncrement();
                this.Update();

                // 2 for framesize field and 1 for table number
                SizeOfQuantizer[3] = FrameSize + 3;

                // Read out the table data
                while(FrameSize > 0)
                {
                  // We are counting down the FrameSize to start the
                                      // stream.
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize--;
                  txtQuantizer4.Text += Top1.ToString() +
                                          Bottom1.ToString() + " ";
                }
                // Update the loading form
                Loading.LoadProgressValue += SizeOfQuantizer[3] - 2;
                Loading.UpdateAndIncrement();
                this.Update();
              }
              else
              {
                // Show an error

              }

              break;
            }
```



```
                case 'c':
                {  // Marker ffdc : Define number of lines, 4 bytes

                  // Read out 4 bytes
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize = SetByteValue(Top1, Bottom1);
                  FrameSize = FrameSize << 8;
                                      // to get the rest of the counter
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize += SetByteValue(Top1, Bottom1);
                  FrameSize = FrameSize << 8;
                                      // to get the rest of the counter
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize += SetByteValue(Top1, Bottom1);
                  FrameSize = FrameSize << 8;
                                      // to get the rest of the counter
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize += SetByteValue(Top1, Bottom1);

                  // Store the number of lines data
                  NumberOfLines = FrameSize;

                  // Update the loading form
                  Loading.LoadProgressValue += 2;
                  Loading.UpdateAndIncrement();
                  this.Update();

                  lblNumberLinesMarker.Text = "ffdc";
                  txtNumberLines.Text = FrameSize.ToString();
                  FrameSize = 0;
                  break;
                }

                case 'd':
                {  // Marker ffdd : Define restart interval, 4 bytes

                  // Read out 4 bytes
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize = SetByteValue(Top1, Bottom1);
                  FrameSize = FrameSize << 8;
                                      // to get the rest of the counter
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize += SetByteValue(Top1, Bottom1);
                  FrameSize = FrameSize << 8;
                                      // to get the rest of the counter
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize += SetByteValue(Top1, Bottom1);
                  FrameSize = FrameSize << 8;
```

```
                                // to get the rest of the counter
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize += SetByteValue(Top1, Bottom1);

                // Store Restart Data
                RestartInterval = FrameSize;

                // Update the loading form
                Loading.LoadProgressValue += 2;
                Loading.UpdateAndIncrement();
                this.Update();

                lblRestartMarker.Text = "ffdd";
                txtRestart.Text = FrameSize.ToString();
                FrameSize = 0;
                break;
              }

              case 'e':
              {  // Marker ffde : Define Hierarchial Progression

                lblHierarchialMarker.Text = "ffde";

                // Read in the Frame Size to set values
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize = SetByteValue(Top1, Bottom1);
                FrameSize = FrameSize << 8;
                                // to get the rest of the counter
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize += SetByteValue(Top1, Bottom1);
                SizeOfProgression = FrameSize;
                FrameSize -= 2;
                                // For the 2 bytes that hold the frame s
ize

                while(FrameSize > 0)
                {
                  // We are counting down the FrameSize to start the stream.
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize--;
                  txtHierarchial.Text +=
                                          Top1.ToString() + Bottom1.ToStri
ng() + " ";
                }

                // Update the loading form
                Loading.LoadProgressValue +=
                                  ((txtHierarchial.Text.Length * 2)/3) -
 2;
                Loading.UpdateAndIncrement();
                this.Update();

                break;
              }

              case 'f':
              {  // Marker ffdf : Expand Reference Images, 3 bytes
```



```
                // Read out 3 bytes
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize = SetByteValue(Top1, Bottom1);
                FrameSize = FrameSize << 8;
                                // to get the rest of the counter
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize += SetByteValue(Top1, Bottom1);
                FrameSize = FrameSize << 8;
                                // to get the rest of the counter
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize += SetByteValue(Top1, Bottom1);

                // Update the loading form
                Loading.LoadProgressValue += 1;
                Loading.UpdateAndIncrement();
                this.Update();

                // Store the data
                ExpandImage = FrameSize;

                lblExpandMarker.Text = "ffdf";
                txtExpand.Text = FrameSize.ToString();
                FrameSize = 0;
                break;
              }

              default:
              {
                txtError.Text +=
                  "\nError: Invalid File Marker Read!! " +
                  "\n\t-- Marker ffd" + Bottom1.ToString()+
                  " was found at byte index: " +
                  ((int)(Count - 4)).ToString();
                txtError.Update();
                break;
              }

            }  // End of: switch(Bottom1)

            break;

          } // End of: case 'd': // marker ffdX


          case 'e': // marker ffeX
          {      // e0 to ef - Reserved for application data

            if(lblApplicationMarker1.Text == "")
                                lblApplicationMarker1.Text = "ffe" + Bottom1;
            else if(lblApplicationMarker2.Text == "")
                                lblApplicationMarker2.Text = "ffe" + Bottom1;
            else if(lblApplicationMarker3.Text == "")
                                lblApplicationMarker3.Text = "ffe" + Bottom1;
            else if(lblApplicationMarker4.Text == "")
                                lblApplicationMarker4.Text = "ffe" + Bottom1;
            else if(lblApplicationMarker5.Text == "")
                                lblApplicationMarker5.Text = "ffe" + Bottom1;
            else if(lblApplicationMarker6.Text == "")
                                lblApplicationMarker6.Text = "ffe" + Bottom1;
```

```
            else if(lblApplicationMarker7.Text == "")
                                lblApplicationMarker7.Text = "ffe" + Bottom1;
            else if(lblApplicationMarker8.Text == "")
                                lblApplicationMarker8.Text = "ffe" + Bottom1;
            else if(lblApplicationMarker9.Text == "")
                                lblApplicationMarker9.Text = "ffe" + Bottom1;
            else if(lblApplicationMarker10.Text == "")
                                lblApplicationMarker10.Text = "ffe" + Bottom1;

            // Read in the Frame Size to set values
            Top1 = HexChars[Count];
            Count++;
            Bottom1 = HexChars[Count];
            Count++;
            FrameSize = SetByteValue(Top1, Bottom1);
            FrameSize = FrameSize << 8;
                              // to get the rest of the counter
            Top1 = HexChars[Count];
            Count++;
            Bottom1 = HexChars[Count];
            Count++;
            FrameSize += SetByteValue(Top1, Bottom1);
            FrameSize -= 2;  // For the 2 bytes that hold the frame size

            // Update the loading form
            Loading.LoadProgressValue += 2;
            Loading.UpdateAndIncrement();
            this.Update();

            if(txtApplicationData1.Text == "")
            {
              SizeOfAppData[0] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtApplicationData1.Text += Top1.ToString() +
                                        Bottom1.ToString() + " ";
              }
              // Update the loading form
              Loading.LoadProgressValue += SizeOfAppData[0] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtApplicationData2.Text == "")
            {
              SizeOfAppData[1] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtApplicationData2.Text += Top1.ToString() +
                                        Bottom1.ToString() + " ";
              }
              // Update the loading form
              Loading.LoadProgressValue += SizeOfAppData[1] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtApplicationData3.Text == "")
            {
```



```
              SizeOfAppData[2] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtApplicationData3.Text += Top1.ToString() +
                                        Bottom1.ToString() + " ";
              }
              // Update the loading form
              Loading.LoadProgressValue += SizeOfAppData[2] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtApplicationData4.Text == "")
            {
              SizeOfAppData[3] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtApplicationData4.Text += Top1.ToString() +
                                        Bottom1.ToString() + " ";
              }
              // Update the loading form
              Loading.LoadProgressValue += SizeOfAppData[3] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtApplicationData5.Text == "")
            {
              SizeOfAppData[4] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtApplicationData5.Text += Top1.ToString() +
                                        Bottom1.ToString() + " ";
              }
              // Update the loading form
              Loading.LoadProgressValue += SizeOfAppData[4] - 2;
              Loading.UpdateAndIncrement();
              this.Update();
            }
            else if(txtApplicationData6.Text == "")
            {
              SizeOfAppData[5] = FrameSize + 2;
              while(FrameSize > 0)
              {
                // We are counting down the FrameSize to start the stream.
                Top1 = HexChars[Count];
                Count++;
                Bottom1 = HexChars[Count];
                Count++;
                FrameSize--;
                txtApplicationData6.Text += Top1.ToString() +
                                        Bottom1.ToString() + " ";
              }
```

```
                // Update the loading form
                Loading.LoadProgressValue += SizeOfAppData[5] - 2;
                Loading.UpdateAndIncrement();
                this.Update();
              }
              else if(txtApplicationData7.Text == "")
              {
                SizeOfAppData[6] = FrameSize + 2;
                while(FrameSize > 0)
                {
                  // We are counting down the FrameSize to start the stream.
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize--;
                  txtApplicationData7.Text += Top1.ToString() +
                                              Bottom1.ToString() + " ";
                }
                // Update the loading form
                Loading.LoadProgressValue += SizeOfAppData[6] - 2;
                Loading.UpdateAndIncrement();
                this.Update();
              }
              else if(txtApplicationData8.Text == "")
              {
                SizeOfAppData[7] = FrameSize + 2;
                while(FrameSize > 0)
                {
                  // We are counting down the FrameSize to start the stream.
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize--;
                  txtApplicationData8.Text += Top1.ToString() +
                                              Bottom1.ToString() + " ";
                }
                // Update the loading form
                Loading.LoadProgressValue += SizeOfAppData[7] - 2;
                Loading.UpdateAndIncrement();
                this.Update();
              }
              else if(txtApplicationData9.Text == "")
              {
                SizeOfAppData[8] = FrameSize + 2;
                while(FrameSize > 0)
                {
                  // We are counting down the FrameSize to start the stream.
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize--;
                  txtApplicationData9.Text += Top1.ToString() +
                                              Bottom1.ToString() + " ";
                }
                // Update the loading form
                Loading.LoadProgressValue += SizeOfAppData[8] - 2;
                Loading.UpdateAndIncrement();
                this.Update();
              }
              else if(txtApplicationData10.Text == "")
              {
                SizeOfAppData[9] = FrameSize + 2;
                while(FrameSize > 0)
                {
                  // We are counting down the FrameSize to start the stream.
                  Top1 = HexChars[Count];
```



```
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize--;
                  txtApplicationData10.Text += Top1.ToString() +
                                               Bottom1.ToString() + " ";
                }
                // Update the loading form
                Loading.LoadProgressValue += SizeOfAppData[9] - 2;
                Loading.UpdateAndIncrement();
                this.Update();
              }

              break;
            }

            case 'f': // marker fffX
            {
              switch(Bottom1)
              {
                case '0': goto case 'd';
                case '1': goto case 'd';
                case '2': goto case 'd';
                case '3': goto case 'd';
                case '4': goto case 'd';
                case '5': goto case 'd';
                case '6': goto case 'd';
                case '7': goto case 'd';
                case '8': goto case 'd';
                case '9': goto case 'd';
                case 'a': goto case 'd';
                case 'b': goto case 'd';
                case 'c': goto case 'd';
                case 'd':
                {  // marker fff0 to fffd: Reserved for JPEG extensions

                  txtError.Text +=
                    "\nPossible Error: Reserved for JPEG Extensions Marker "+
                    "Found!!\n\t-- Marker ff" + Top1.ToString() +
                                               Bottom1.ToString() + " was found at byte
 index: " +
                    ((int)(Count - 4)).ToString();
                  txtError.Update();
                  break;
                }

                case 'e': // marker fffe - Comments
                {
                  // Read in the Frame Size to set values
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize = SetByteValue(Top1, Bottom1);
                  FrameSize = FrameSize << 8;
                                               // to get the rest of the counter
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize += SetByteValue(Top1, Bottom1);
                  SizeOfComments = FrameSize;
                  FrameSize -= 2;
                                               // For the 2 bytes that hold the frame s
ize

                  // Update the loading form
                  Loading.LoadProgressValue += 2;
                  Loading.UpdateAndIncrement();
```

```
                this.Update();

                while(FrameSize > 0)
                {
                  // We are counting down the FrameSize to start the stream.
                  Top1 = HexChars[Count];
                  Count++;
                  Bottom1 = HexChars[Count];
                  Count++;
                  FrameSize--;
                  Temp = SetByteValue(Top1, Bottom1);
                  txtComments.Text += (char)Temp;
                }

                // Update the loading form
                Loading.LoadProgressValue += txtComments.Text.Length - 2;
                Loading.UpdateAndIncrement();
                this.Update();

                break;

              }
              case 'f': // marker ffff -- Marker Not Defined
              {
                txtError.Text +=
                  "\nError: Marker NOT defined " +
                  "\n\t-- Marker ffff was found at byte index: " +
                  ((int)(Count - 4)).ToString();
                txtError.Update();
                break;
              }

              default:
              {
                txtError.Text +=
                  "\nError: Invalid File Marker Read!! " +
                  "\n\t-- Marker ffd" + Bottom1.ToString()+
                  " was found at byte index: " +
                  ((int)(Count - 4)).ToString();
                txtError.Update();
                break;
              }

            } // End of: switch(Bottom1)

            break;
          }

          default:
          {
            txtError.Text +=
              "\nError: Invalid File Marker Read!! " +
              "\n\t-- Marker ff" + Top1.ToString() + Bottom1.ToString()+
              " was found at byte index: " +
              ((int)(Count - 4)).ToString();
            txtError.Update();
            break;
          }

        }  // End of: switch(Top1)

      }  // End of: if(Top1 == 'f' && Bottom1 == 'f')
      else
      {
        if(ShowWarning(
          "\nInvalid File Marker Read!" +
          "\nImage maybe damaged or image may not be properly fromatted "+
                      "to be a JPEG.\n\nLoad Operation Cancelled!"))
        {
```



```
            txtError.Text +=
              "\nError: Invalid Marker Found!! " +
              "\n\t-- Marker ff" + Top1.ToString() + Bottom1.ToString() +

              " was found.";

            txtError.Update();
            ShowWarning(
              "\nLoad Operation was canceled" +
              "\nImage maybe damaged or image may not be properly fromatted"+
              " to be a JPEG.");
            break;
          }

        }

      } // End of: while(Count < FileLeng)

      Loading.Dispose();

    } // End of: private void LoadInterfaceData(ref jfile HexChars)



    #endregion Methods to Convert from Binary to ACSII


    #region Methods to Convert from ACSII to Binary


    /// <summary>
    /// Thie Method is used to check if a char value is a valid Hexadecimal
    /// char value.  The method returns TRUE if the char is '0' to '9' or
    /// if it 'a' to 'f' (also 'A' to 'F'), otherwise FALSE is returned.
    /// </summary>
    /// <param name="HexValue">The CHAR value to check.</param>
    /// <returns>Returns TRUE if the char is '0' to '9' or if it 'a'
    /// to 'f' (also 'A' to 'F'), otherwise FALSE is returned.</returns>
    private bool IsValidHex(char HexValue)
    {
      if(HexValue == '0' || HexValue == '1' || HexValue == '2' ||
        HexValue == '3' || HexValue == '4' || HexValue == '5' ||
        HexValue == '6' || HexValue == '7' || HexValue == '8' ||
        HexValue == '9')
      {
        return true;
      }
      else
      {
        HexValue = Char.ToLower(HexValue);
        if(HexValue == 'a' || HexValue == 'b' || HexValue == 'c' ||
          HexValue == 'd' || HexValue == 'e' || HexValue == 'f')
        {
          return true;
        }
        else return false;
      }
    }


    /// <summary>
    /// Pre-conditions:    None.
    /// Post-conditions:
    ///    The LowBits and HighBits parameters are converted to integers and
    ///    then combined to form the byte value that is returned by this
    ///    function.
    ///  Description:
    ///    The purpose of this method is to allow the caller to easily
    ///    convert two ASCII characters, between 0 to F, to their binary
```

```
    ///    values and then combine them to form a one-byte value.  This
    ///    function should call the Convert() method that takes a char and
    ///    returns a byte for each of these two parameters to get the
    ///    integer value of each.  Then, it should combine both of these
    ///    integer values to form one full byte value.  Finally, this byte
    ///    value should be returned when the function exits.
    /// </summary>
    /// <param name="HighBits">The HighBits parameter is an ASCII character
    /// that represents a value of 0 to 15, in the form of 0 to F, for the 4
    /// most significant bits of the byte that will be returned.</param>
    /// <param name="LowBits">The LowBits parameter is an ASCII character
    /// that represents a value of 0 to 15, in the form of 0 to F, for the 4
    /// least significant bits of the byte that will be returned.</param>
    /// <returns>Function returns a byte value based upon the parameters
    /// passed in.</returns>
    private byte SetByteValue(char HighBits, char LowBits)
    {
      High = Convert(HighBits);  // Get 4 high bits
      High = High << 4;      // Shift up 4 bits
      High += Convert(LowBits);  // Add on the lower bits
      return (byte)High;
    }


    /// <summary>
    /// Pre-conditions:    None.
    /// Post-conditions:
    ///    An integer representing the binary value of the hexadecimal ASCII
    ///    character parameter passed will be returned.
    ///  Description:
    ///    The purpose of this function allows the caller to convert an ASCII
    ///    character between 0 and F to its corresponding integer value of 0
    ///    to 15.  This function will return a M-^V1 if the char parameter
    ///    passed in is not between the value of 0 and F and an error
    ///    message will be displayed for the user.
    /// </summary>
    /// <param name="Hex">The Hex parameter is an ASCII character between 0
    /// and F.</param>
    /// <returns>Function returns an int based upon the hexadecimal value of
    /// the char parameter.</returns>
    private int Convert(char Hex)
    {
      switch(Hex.ToString().ToLower()[0])
      {
        case '0': return 0;
        case '1': return 1;
        case '2': return 2;
        case '3': return 3;
        case '4': return 4;
        case '5': return 5;
        case '6': return 6;
        case '7': return 7;
        case '8': return 8;
        case '9': return 9;
        case 'a': return 10;
        case 'b': return 11;
        case 'c': return 12;
        case 'd': return 13;
        case 'e': return 14;
        case 'f': return 15;
        default:
        {
          ShowWarning(
            "Function \"int Convert(char);\" encountered an unrecognized " +
                        "character!!\nThis is a SERIOUS error! Please inform dev
eloper.");
          return -1;
        }
      }
```



```
    }


    /// <summary>
    /// Pre-conditions:    None.
    /// Post-conditions:
    ///    All of the character data contained in each of the data TextBox
    ///    controls for the JPEG file is recombined and input, in order, into
    ///    the File parameter passed.
    ///  Description:
    ///    The purpose of this method is to take all of the data currently
    ///    loaded in the ManipulatorM-^Rs interface and recombine these values
    ///    into one large byte array.  This byte array will contain all of the
    ///    binary data in the exact form the as the current ASCII chars loaded
    ///    in the data fields of the Manipulator.  As such, this function is
    ///    one of the largest functions in the Manipulator and performs many
    ///    tasks during its execution.  This function should start dequeuing
    ///    and re-enqueuing the markers stored in the FileOrder Queue.  For
    ///    each file marker found in this queue, the data in the corresponding
    ///    interface data TextBox should be processed.  This function should
    ///    read the data from the particular TextBox, convert this data to
    ///    binary and then input the resulting data into the File byte array
    ///    parameter passed into this function.  Lastly, this method should do
    ///    lots error checking to make sure this function executes properly.
    ///    If an error is encountered, then the ShowWarning() method should be
    ///    called to display the error to the user and the txtError TextBox
    ///    control should be updated with this error information.
    /// </summary>
    /// <param name="File">The File parameter is storage space for the new
    /// file byte array.  All the data for the new JPEG image will be based on
    /// the conversion of the ASCII characters that are currently loaded in
    /// all of the data fields of the ManipulatorM-^Rs interface.</param>
    /// <returns>Function returns True if it completes successfully, else
    /// False.</returns>
    private bool CreateManipulatedPicture(ref byte[] File)
    {  // Returns true if completed correctly.
      try
      {
        Loading = new frmLoad();

        char A = 'f', B = 'f', C = 'X', D = 'X';
        int count = 0, HuffmanNumber = 0, QuantizerNumber = 0;
                int AppDataNumber = 0;

        if(File != null) File = null;
        File = new byte[MAX_BYTES];

        Loading.StartLoading(0, FileSize, 1);

        while(A == 'f' && B == 'f')
        {
          A = (char)FileOrder.Dequeue();
          B = (char)FileOrder.Dequeue();
          C = (char)FileOrder.Dequeue();
          D = (char)FileOrder.Dequeue();
          FileOrder.Enqueue(A);
          FileOrder.Enqueue(B);
          FileOrder.Enqueue(C);
          FileOrder.Enqueue(D);

          NewData[count] = SetByteValue(A, B);
          count++;
          NewData[count] = SetByteValue(C, D);
          count++;

          // Update the loading form
          if(Loading.Canceled)
          {
            Loading.Dispose();
```

```
            return false;
          }
          Loading.UpdateAndIncrement();
          Loading.UpdateAndIncrement();
          this.Update();

          // If we are at the end of the file, we'll break
          if(A == 'f' && B == 'f' && C == 'd' && D == '9') break;

          if(A == 'f' && B == 'f')
          {
            switch(C)
            {  // JPEG FILE MARKERS, Pg 106 in "JPEG" by:
                   // Pennebaker & Mitchell

              case '0': // Marker ff0X
              {
                switch(D)
                {
                  case '0': // Marker ff00 - Marker Not Defined
                  {
                    txtError.Text +=
                      "\nError: Marker NOT defined " +
                      "\n\t-- Marker FF00 was found in the original file" +
                                        " stream! Marker and data NOT written
to new file.";
                    txtError.Update();
                    break;
                  }
                  case '1': // Marker ff01
                  {
                    txtError.Text +=
                      "\nError: Marker found Temporary use for Arthmetic" +
                                     " Encoding\n\t-- Marker FF01 was found in
the " +
                                        "original file stream. Marker and data
 NOT written " +
                      "to new file.";
                    txtError.Update();

                    break;
                  }
                  case '2': goto case 'f';
                  case '3': goto case 'f';
                  case '4': goto case 'f';
                  case '5': goto case 'f';
                  case '6': goto case 'f';
                  case '7': goto case 'f';
                  case '8': goto case 'f';
                  case '9': goto case 'f';
                  case 'a': goto case 'f';
                  case 'b': goto case 'f';
                  case 'c': goto case 'f';
                  case 'd': goto case 'f';
                  case 'e': goto case 'f';
                  case 'f':
                  {
                    // Marker ff02 to ff0f - Reserved
                    txtError.Text +=
                      "\nError: Reserved Marker Found!! " +
                      "\n\t-- Marker ff0" + D.ToString()+
                      " was found in the original stream. " +
                                        " Marker and data NOT written to new f
ile.";
                    txtError.Update();
                    break;
                  }

                  default:
```



```
                  {
                    txtError.Text +=
                      "\nError: Invalid File Marker Read!! " +
                      "\n\t-- Marker ff0" + D.ToString()+
                      " was found in the original stream. " +
                                        " Marker and data NOT written to new f
ile.";
                    txtError.Update();
                    break;
                  }

                } // End of: switch(D)

                break;

              } // End of: case '0';

              case '1': goto case 'b';
              case '2': goto case 'b';
              case '3': goto case 'b';
              case '4': goto case 'b';
              case '5': goto case 'b';
              case '6': goto case 'b';
              case '7': goto case 'b';
              case '8': goto case 'b';
              case '9': goto case 'b';
              case 'a': goto case 'b';
              case 'b':
              { // Marker ff10 to ffbf - Reserved
                txtError.Text +=
                  "\nError: Reserved Marker Found!! " +
                  "\n\t-- Marker ff" + C.ToString() + D.ToString()+
                  " was found in the original stream. " +
                                    " Marker and data NOT written to new file.";
                txtError.Update();
                break;
              }

              case 'c': // marker ffcX - huffman tables
              {
                bool Read = true;

                switch(D)
                {
                  // Start of: Nondifferential Huffman-Coding Frames
                  case '0': // marker ffc0 - Baseline DCT
                  {
                    // Manipulated 01-18-2004
                    // HeaderSize = 2 because 2 bytes for size field
                    int HeaderSize = 2;
                    char Top, Bottom;
                    byte [] HeaderData = new byte[100];

                    // Set Precision - 1 Byte
                    txtPrecision.Text = txtPrecision.Text.Trim();
                    if(txtPrecision.Text.Length < 2)
                    {
                      ShowWarning("The Precision on the Headers Tab, must" +
                                                "be EXACTLY 1 bytes!\n" +
                        "Random values will be added to solve this problem!",
                        "Warning, image data altered!");
                      txtPrecision.Text = "00";
                    }
                    Top = txtPrecision.Text[0];
                    Bottom = txtPrecision.Text[1];
                    HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                    HeaderSize++;

                    // Update the loading form
```

```
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // Set Number Lines - 2 Bytes
                    txtNumberHuffmanLines.Text =
                                        txtNumberHuffmanLines.Text.Trim(
);
                    if(txtNumberHuffmanLines.Text.Trim().Length < 5)
                    {
                      ShowWarning("The number of Lines on the Headers Tab, "+
                                        "must be EXACTLY 2 bytes!\n" +
                        "Random values will be added to solve this problem!",
                        "Warning, image data altered!");
                        txtNumberHuffmanLines.Text = "00 00";
                    }
                    Top = txtNumberHuffmanLines.Text[0];
                    Bottom = txtNumberHuffmanLines.Text[1];
                    HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                    HeaderSize++;
                    Top = txtNumberHuffmanLines.Text[3];
                    Bottom = txtNumberHuffmanLines.Text[4];
                    HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                    HeaderSize++;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // Set Number of samples per line - 2 Bytes
                    txtNumberHuffmanSamples.Text =
                                        txtNumberHuffmanSamples.Text.Tri
m();
                    if(txtNumberHuffmanSamples.Text.Trim().Length < 5)
                    {
                      ShowWarning("The number of Samples per Line on the
                                         Headers Tab, must be EXACTLY 2
 bytes!\n" +
                        "Random values will be added to solve this problem!",
                        "Warning, image data altered!");
                      txtNumberHuffmanSamples.Text = "00 00";
                    }
                    Top = txtNumberHuffmanSamples.Text[0];
                    Bottom = txtNumberHuffmanSamples.Text[1];
                    HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                    HeaderSize++;
                    Top = txtNumberHuffmanSamples.Text[3];
                    Bottom = txtNumberHuffmanSamples.Text[4];
                    HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                    HeaderSize++;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // Get number of image components - 1 Byte
                    txtNumberImageComponents.Text =
                                        txtNumberImageComponents.Text.Tr
im();
                    if(txtNumberImageComponents.Text.Length < 2)
                    {
                      ShowWarning("The Number of Image Components will be
                                         calculated!\n",
                        "Warning, image data altered!");
                      txtNumberImageComponents.Text = "00";
                    }
                    Top = txtPrecision.Text[0];
                    Bottom = txtPrecision.Text[1];
```



```
                    HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                    HeaderSize++;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    this.Update();

                    int k = 0;

                    // Get rid of "Identifier, Horizontal, Vertical, Q-Table: \n
" at
                    // the beginning of the control.
                    string CData = txtComponents.Text.ToString();
                    while(CData[k] != '\n') k++;
                    k++;

                    // Get all the component data
                    CData = CData.Substring(k,
                                          (txtComponents.Text.Length - k))
;
                    k = 0;

                    // Get all of the components
                    byte NewSize = 0;
                    int SizeIndex = HeaderSize - 1;
                    bool Done = false;

                    while(k < CData.Length)
                    {
                      // Move to the next data
                      while((CData[k] == ' ' || CData[k] == ','
                           || CData[k] == '\n' || CData[k] == '\t')
                           && (k < CData.Length))
                      {
                        k++;
                        if(!(k < CData.Length))
                        {
                          Done = true;
                          break;
                        }

                      }
                      if(Done) break;

                      // Get Component identifier - 1 byte
                      Top = CData[k];
                      k++;
                      Bottom = CData[k];
                      k++;
                      HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                      HeaderSize++;

                      while((CData[k] == ' ' || CData[k] == ','
                        || CData[k] == '\n' || CData[k] == '\t')
                        && (k < CData.Length))
                      {
                        k++;
                        if(!(k < CData.Length))
                        {
                          Done = true;
                          break;
                        }
                      }
                      if(Done)
                      {
                        Bottom = '0';
                        HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                        HeaderSize++;
                        NewSize++;
```

```
                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    this.Update();

                    break;
                  }

                  // Get Horizontal and Vertical Sampling factor - 4 bits ea
ch, or
                  Top = CData[k];
                  k++;

                  // For Horizontal and Vertical Sampling factor - 4 bits ea
ch
                  while((CData[k] == ' ' || CData[k] == ','
                    || CData[k] == '\n' || CData[k] == '\t')
                    && (k < CData.Length))
                  {
                    k++;
                    if(!(k < CData.Length))
                    {
                      Done = true;
                      break;
                    }
                  }
                  if(Done)
                  {
                    Bottom = '0';
                    HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                    HeaderSize++;
                    NewSize++;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    this.Update();

                    break;
                  }

                  Bottom = CData[k];
                  k++;
                  HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                  HeaderSize++;

                  while((CData[k] == ' ' || CData[k] == ','
                    || CData[k] == '\n' || CData[k] == '\t')
                    && (k < CData.Length))
                  {
                    k++;
                    if(!(k < CData.Length))
                    {
                      Done = true;
                      break;
                    }
                  }
                  if(Done)
                  {
                    Bottom = '0';
                    HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                    HeaderSize++;
                    NewSize++;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    this.Update();

                    break;
                  }
```



```
                  // Get Quantization Table Selector - 1 byte
                  Top = CData[k];
                  k++;
                  Bottom = CData[k];
                  k++;
                  HeaderData[HeaderSize] = SetByteValue(Top, Bottom);
                  HeaderSize++;

                  // Update the loading form
                  Loading.UpdateAndIncrement();
                  this.Update();
                  NewSize++;
                }

                // Set the new Number of Components
                HeaderData[SizeIndex] = NewSize;

                // Set the new Header Frame size
                HeaderData[0] = (byte)((HeaderSize >> 8) % 256);
                HeaderData[1] = (byte)(HeaderSize % 256);

                // Now copy the HeaderData
                for(int i = 0; i < HeaderSize; i++)
                {
                  NewData[count] = HeaderData[i];
                  count++;
                }

                Read = false; // Skip reading values at end of loop

                // End of change
                break;
              }
              case '1': // marker ffc1 - Extended Sequential DCT
              {
                // Implemented generically in this version
                break;
              }
              case '2': // marker ffc2 - Progressive DCT
              {
                // Implemented generically in this version
                break;
              }
              case '3': // marker ffc3 - Lossless (Sequential)
              {
                // Implemented generically in this version
                break;
              }
              // End of: Nondifferential Huffman-Coding Frames


              case '4': // marker ffc4 - Define Huffman Marker
              {
                // Implemented generically in this version
                break;
              }


              // Start of: Differential Huffman-Coding Frames
              case '5': // marker ffc5 - Differential Sequential DCT
              {
                // Implemented generically in this version
                break;
              }
              case '6': // marker ffc6 - Differential Progressive DCT
              {
                // Implemented generically in this version
                break;
```

```
                }
                case '7': // marker ffc7 - Differential Lossless
                {
                  // Implemented generically in this version
                  break;
                }
                // End of: Differential Huffman-Coding Frames

                case '8': // marker ffc8 - Reserved for JPEG Extensions
                {
                  txtError.Text +=
                    "\nError: Reserved For JPEG Extensions Marker Found!!"+
                    "\n\t-- Marker ffcd" +
                    " was found in the original file stream." +
                                  "\nMarker and data not written to the
new file.";
                  txtError.Update();
                  Read = false; // Skip reading values for this marker
                  break;
                }


                // Start of: Nondifferential Arithmetic-Coding Frames
                case '9': // marker ffc9 - Exteneded Sequential DCT
                {
                  // Implemented generically in this version
                  break;
                }
                case 'a': // marker ffca - Progressive DCT
                {
                  // Implemented generically in this version
                  break;
                }
                case 'b': // marker ffcb - Lossless (Sequential)
                {
                  // Implemented generically in this version
                  break;
                }
                // End of: Nondifferential Arithmetic-Coding Frames


                case 'c': // marker ffcc -
                          //Define Arithmetic Conditioning Tables
                {
                  // Implemented generically in this version
                  break;
                }


                // Start of: Differential Arithmetic-Coding Frames
                case 'd': // marker ffcd - Differential Sequential DCT
                {
                  // Implemented generically in this version
                  break;
                }
                case 'e': // marker ffce - Differential Progressive DCT
                {
                  // Implemented generically in this version
                  break;
                }
                case 'f': // marker ffcf - Differential Lossless
                {
                  // Implemented generically in this version
                  break;
                }
                // End of: Differential Arithmetic-Coding Frames

                default:
                {
```



```
                  txtError.Text +=
                    "\nError: Invalid File Marker Read!! " +
                    "\n\t-- Marker ffc" + D.ToString()+
                    " was found in original file stream. " +
                    "Marker and data not written to new file.";
                  break;
                }

              }  // End of: switch(D)

              if(Loading.Canceled)
              {
                Loading.Dispose();
                return false;
              }

              if(Read)
              {
                byte Byte1, Byte2;
                int SizeIndex = count;

                // Move ahead of the size field
                count++;
                count++;

                if(HuffmanNumber == 0)
                {
                  int t;
                  string NewHuff = "";
                  char Nibble;

                  // Update the table we're reading
                  HuffmanNumber++;

                  // Read out the content of the TextBox and
                  // check to get only the valid HEX value chars
                  for(int x = 0; x < txtHuffman1.Text.Length; x++)
                  {
                    Nibble = txtHuffman1.Text[x];
                    if(IsValidHex(Nibble))
                      NewHuff += Nibble.ToString();
                  }

                  // Check to make sure the size of the new
                  // huffman table is correct and if not, fix
                  if((NewHuff.Length % 2) == 1)
                    NewHuff += "0";

                  // Recalculated the size of the field and
                  // write back to the new file string
                  // for the 2 bytes of size
                  t = (NewHuff.Length + 4)/2;
                  Byte2 = (byte)(t % 256);
                  t >>= 8;
                  Byte1 = (byte)(t % 256);
                  NewData[SizeIndex] = Byte1;
                  SizeIndex++;
                  NewData[SizeIndex] = Byte2;

                  // Update the loading form
                  Loading.UpdateAndIncrement();
                  Loading.UpdateAndIncrement();
                  this.Update();

                  // Now write the new huffman table to the
                  // NewData string.
                  for(int x = 0; x < NewHuff.Length; x+=2)
                  {
                    NewData[count] = SetByteValue(
```

```
            NewHuff[x], NewHuff[x+1]);
          count++;
          Loading.UpdateAndIncrement();
          this.Update();
        }
      }
      else if(HuffmanNumber == 1)
      {
        int t;
        string NewHuff = "";
        char Nibble;

        // Update the table we're reading
        HuffmanNumber++;

        // Read out the content of the TextBox and
        // check to get only the valid HEX value chars
        for(int x = 0; x < txtHuffman2.Text.Length; x++)
        {
          Nibble = txtHuffman2.Text[x];
          if(IsValidHex(Nibble))
            NewHuff += Nibble.ToString();
        }

        // Check to make sure the size of the new
        // huffman table is correct and if not, fix
        if((NewHuff.Length % 2) == 1)
          NewHuff += "0";

        // Recalculated the size of the field and
        // write back to the new file string
        // for the 2 bytes of size
        t = (NewHuff.Length + 4)/2;
        Byte2 = (byte)(t % 256);
        t >>= 8;
        Byte1 = (byte)(t % 256);
        NewData[SizeIndex] = Byte1;
        SizeIndex++;
        NewData[SizeIndex] = Byte2;

        // Update the loading form
        Loading.UpdateAndIncrement();
        Loading.UpdateAndIncrement();
        this.Update();

        // Now write the new huffman table to the
        // NewData string.
        for(int x = 0; x < NewHuff.Length; x+=2)
        {
          NewData[count] = SetByteValue(
            NewHuff[x], NewHuff[x+1]);
          count++;
          Loading.UpdateAndIncrement();
          this.Update();
        }
      }
      else if(HuffmanNumber == 2)
      {
        int t;
        string NewHuff = "";
        char Nibble;

        // Update the table we're reading
        HuffmanNumber++;

        // Read out the content of the TextBox and
        // check to get only the valid HEX value chars
        for(int x = 0; x < txtHuffman3.Text.Length; x++)
        {
```



```
          Nibble = txtHuffman3.Text[x];
          if(IsValidHex(Nibble))
            NewHuff += Nibble.ToString();
        }

        // Check to make sure the size of the new
        // huffman table is correct and if not, fix
        if((NewHuff.Length % 2) == 1)
          NewHuff += "0";

        // Recalculated the size of the field and
        // write back to the new file string
        // for the 2 bytes of size
        t = (NewHuff.Length + 4)/2;
        Byte2 = (byte)(t % 256);
        t >>= 8;
        Byte1 = (byte)(t % 256);
        NewData[SizeIndex] = Byte1;
        SizeIndex++;
        NewData[SizeIndex] = Byte2;

        // Update the loading form
        Loading.UpdateAndIncrement();
        Loading.UpdateAndIncrement();
        this.Update();

        // Now write the new huffman table to the
        // NewData string.
        for(int x = 0; x < NewHuff.Length; x+=2)
        {
          NewData[count] = SetByteValue(
            NewHuff[x], NewHuff[x+1]);
          count++;
          Loading.UpdateAndIncrement();
          this.Update();
        }
      }
      else if(HuffmanNumber == 3)
      {
        int t;
        string NewHuff = "";
        char Nibble;

        // Update the table we're reading
        HuffmanNumber++;

        // Read out the content of the TextBox and
        // check to get only the valid HEX value chars
        for(int x = 0; x < txtHuffman4.Text.Length; x++)
        {
          Nibble = txtHuffman4.Text[x];
          if(IsValidHex(Nibble))
            NewHuff += Nibble.ToString();
        }

        // Check to make sure the size of the new
        // huffman table is correct and if not, fix
        if((NewHuff.Length % 2) == 1)
          NewHuff += "0";

        // Recalculated the size of the field and
        // write back to the new file string
        // for the 2 bytes of size
        t = (NewHuff.Length + 4)/2;
        Byte2 = (byte)(t % 256);
        t >>= 8;
        Byte1 = (byte)(t % 256);
        NewData[SizeIndex] = Byte1;
        SizeIndex++;
```

```
      NewData[SizeIndex] = Byte2;

      // Update the loading form
      Loading.UpdateAndIncrement();
      Loading.UpdateAndIncrement();
      this.Update();

      // Now write the new huffman table to the
      // NewData string.
      for(int x = 0; x < NewHuff.Length; x+=2)
      {
        NewData[count] = SetByteValue(
          NewHuff[x], NewHuff[x+1]);
        count++;
        Loading.UpdateAndIncrement();
        this.Update();
      }
    }
    else if(HuffmanNumber == 4)
    {
      int t;
      string NewHuff = "";
      char Nibble;

      // Update the table we're reading
      HuffmanNumber++;

      // Read out the content of the TextBox and
      // check to get only the valid HEX value chars
      for(int x = 0; x < txtHuffman5.Text.Length; x++)
      {
        Nibble = txtHuffman5.Text[x];
        if(IsValidHex(Nibble))
          NewHuff += Nibble.ToString();
      }

      // Check to make sure the size of the new
      // huffman table is correct and if not, fix
      if((NewHuff.Length % 2) == 1)
        NewHuff += "0";

      // Recalculated the size of the field and
      // write back to the new file string
      // for the 2 bytes of size
      t = (NewHuff.Length + 4)/2;
      Byte2 = (byte)(t % 256);
      t >>= 8;
      Byte1 = (byte)(t % 256);
      NewData[SizeIndex] = Byte1;
      SizeIndex++;
      NewData[SizeIndex] = Byte2;

      // Update the loading form
      Loading.UpdateAndIncrement();
      Loading.UpdateAndIncrement();
      this.Update();

      // Now write the new huffman table to the
      // NewData string.
      for(int x = 0; x < NewHuff.Length; x+=2)
      {
        NewData[count] = SetByteValue(
          NewHuff[x], NewHuff[x+1]);
        count++;
        Loading.UpdateAndIncrement();
        this.Update();
      }
    }
    else if(HuffmanNumber == 5)
```



```
    {
      int t;
      string NewHuff = "";
      char Nibble;

      // Update the table we're reading
      HuffmanNumber++;

      // Read out the content of the TextBox and
      // check to get only the valid HEX value chars
      for(int x = 0; x < txtHuffman6.Text.Length; x++)
      {
        Nibble = txtHuffman6.Text[x];
        if(IsValidHex(Nibble))
          NewHuff += Nibble.ToString();
      }

      // Check to make sure the size of the new
      // huffman table is correct and if not, fix
      if((NewHuff.Length % 2) == 1)
        NewHuff += "0";

      // Recalculated the size of the field and
      // write back to the new file string
      // for the 2 bytes of size
      t = (NewHuff.Length + 4)/2;
      Byte2 = (byte)(t % 256);
      t >>= 8;
      Byte1 = (byte)(t % 256);
      NewData[SizeIndex] = Byte1;
      SizeIndex++;
      NewData[SizeIndex] = Byte2;

      // Update the loading form
      Loading.UpdateAndIncrement();
      Loading.UpdateAndIncrement();
      this.Update();

      // Now write the new huffman table to the
      // NewData string.
      for(int x = 0; x < NewHuff.Length; x+=2)
      {
        NewData[count] = SetByteValue(
          NewHuff[x], NewHuff[x+1]);
        count++;
        Loading.UpdateAndIncrement();
        this.Update();
      }
    }
    else if(HuffmanNumber == 6)
    {
      int t;
      string NewHuff = "";
      char Nibble;

      // Update the table we're reading
      HuffmanNumber++;

      // Read out the content of the TextBox and
      // check to get only the valid HEX value chars
      for(int x = 0; x < txtHuffman7.Text.Length; x++)
      {
        Nibble = txtHuffman7.Text[x];
        if(IsValidHex(Nibble))
          NewHuff += Nibble.ToString();
      }

      // Check to make sure the size of the new
      // huffman table is correct and if not, fix
```

```
                    if((NewHuff.Length % 2) == 1)
                      NewHuff += "0";

                    // Recalculated the size of the field and
                    // write back to the new file string
                    // for the 2 bytes of size
                    t = (NewHuff.Length + 4)/2;
                    Byte2 = (byte)(t % 256);
                    t >>= 8;
                    Byte1 = (byte)(t % 256);
                    NewData[SizeIndex] = Byte1;
                    SizeIndex++;
                    NewData[SizeIndex] = Byte2;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // Now write the new huffman table to the
                    // NewData string.
                    for(int x = 0; x < NewHuff.Length; x+=2)
                    {
                      NewData[count] = SetByteValue(
                        NewHuff[x], NewHuff[x+1]);
                      count++;
                      Loading.UpdateAndIncrement();
                      this.Update();
                    }
                  }
                  else if(HuffmanNumber == 7)
                  {
                    int t;
                    string NewHuff = "";
                    char Nibble;

                    // Update the table we're reading
                    HuffmanNumber++;

                    // Read out the content of the TextBox and
                    // check to get only the valid HEX value chars
                    for(int x = 0; x < txtHuffman8.Text.Length; x++)
                    {
                      Nibble = txtHuffman8.Text[x];
                      if(IsValidHex(Nibble))
                        NewHuff += Nibble.ToString();
                    }

                    // Check to make sure the size of the new
                    // huffman table is correct and if not, fix
                    if((NewHuff.Length % 2) == 1)
                      NewHuff += "0";

                    // Recalculated the size of the field and
                    // write back to the new file string
                    // for the 2 bytes of size
                    t = (NewHuff.Length + 4)/2;
                    Byte2 = (byte)(t % 256);
                    t >>= 8;
                    Byte1 = (byte)(t % 256);
                    NewData[SizeIndex] = Byte1;
                    SizeIndex++;
                    NewData[SizeIndex] = Byte2;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    Loading.UpdateAndIncrement();
                    this.Update();
```



```
                    // Now write the new huffman table to the
                    // NewData string.
                    for(int x = 0; x < NewHuff.Length; x+=2)
                    {
                      NewData[count] = SetByteValue(
                        NewHuff[x], NewHuff[x+1]);
                      count++;
                      Loading.UpdateAndIncrement();
                      this.Update();
                    }
                  }
                  else
                  {
                    txtError.Text +=
                      "\nError: Too Many Huffman Tables!! " +
                      "\n\t-- Marker ff" + C.ToString() + D.ToString()+
                      " was found in the original stream. " +
                                          " Marker and data NOT written to new f
ile.";
                    txtError.Update();
                    return false;
                  }

                } // End of: if(Read);
                else
                {
                  Read = true;
                }

                break;
              }  // End of: case 'c': // marker ffcX

              case 'd': // marker ffdX
              {
                switch(D)
                {
                  case '0': goto case '7';
                  case '1': goto case '7';
                  case '2': goto case '7';
                  case '3': goto case '7';
                  case '4': goto case '7';
                  case '5': goto case '7';
                  case '6': goto case '7';
                  case '7':
                  {  // Marker ffd0 to ffd7

                    if(Loading.Canceled)
                    {
                      Loading.Dispose();
                      return false;
                    }

                    string NewValue = "";
                    char Nibble;
                    for(int x = 0; x < txtRestartMod8.Text.Length; x += 3)
                    {
                      // Check to make sure the values are correct
                      Nibble = txtRestartMod8.Text[x];
                      if(IsValidHex(Nibble))
                        NewValue += Nibble.ToString();
                    }

                    // Make sure the new length is long enough
                    if(NewValue.Length < 4)
                      NewValue += "0" + "0" + "0" + "0";

                    // Write the new values to the NewData
                    for(int x = 0; x < 4; x += 2)
                    {
```

```
        NewData[count] = SetByteValue(
          NewValue[x], NewValue[x+1]);
        count++;

        // Update the loading form
        Loading.UpdateAndIncrement();
        this.Update();
      }

      break;
    }

    case '8':
    {  // Marker ffd8 : Start of Image
      break;
    }

    case '9':
    {  // Marker ffd9 : End of image
      // Covered by: case ffda
      break;
    }

    case 'a':
    {  // Marker ffda : Start of Scan

      byte Byte1, Byte2;
      int SizeIndex = count;
      int t;

      // Check for loading canceled
      if(Loading.Canceled)
      {
        Loading.Dispose();
        return false;
      }

      // Move past the size field
      count++;
      count++;

      char Nibble;
      string NewScan = "";

      // Get Scan Header
      for(int x = 0; x < txtScanHeader.Text.Length; x++)
      {
        Nibble = txtScanHeader.Text[x];
        if(IsValidHex(Nibble))
          NewScan += Nibble.ToString();
      }

      // Check to make sure the new size is valid
      if((NewScan.Length % 2) == 1)
        NewScan += "0";

      // Calculate new Scan Header size
      t = ((NewScan.Length + 4)/2);
      Byte2 = (byte)(t % 256);
      t >>= 8;
      Byte1 = (byte)(t % 256);
      NewData[SizeIndex] = Byte1;
      SizeIndex++;
      NewData[SizeIndex] = Byte2;

      // Update Loading Form
      Loading.UpdateAndIncrement();
      Loading.UpdateAndIncrement();
      this.Update();
```



```
      // Write the new Scan Header to NewData
      for(int x = 0; x < NewScan.Length; x += 2)
      {
        NewData[count] = SetByteValue(
          NewScan[x], NewScan[x+1]);
        count++;

        Loading.UpdateAndIncrement();
        this.Update();
      }

      // Check for loading canceled
      if(Loading.Canceled)
      {
        Loading.Dispose();
        return false;
      }

      // Get Encoded Stream
      //
      // UNSAFE - These values ARE ASSUMED VALID
      // since they cannot be altered by the interface
      for(int x = 0; x < EncodedData.Length; x += 2)
      {
        NewData[count] = SetByteValue(
          EncodedData[x], EncodedData[x+1]);
        count++;

        // Check for loading canceled
        if(Loading.Canceled)
        {
          Loading.Dispose();
          return false;
        }
        else
        {
          Loading.UpdateAndIncrement();
          this.Update();
        }
      }
      break;
    }

    case 'b':
    {  // Marker ffdb : Define Quantization Table

      byte Byte1;
      byte Byte2;
      int SizeIndex = count;
      int t;
      string NewQuant = "";
      char Nibble;

      // Check for loading canceled
      if(Loading.Canceled)
      {
        Loading.Dispose();
        return false;
      }

      // Move past the size field
      count++;
      count++;

      if(QuantizerNumber == 0)
      {
        // Update the table we're reading
        QuantizerNumber++;
```

```
                    // Get the table number
                    if(txtQuantizerTableNum1.Text.Length < 2)
                      txtQuantizerTableNum1.Text = "0" + "0";
                    Nibble = txtQuantizerTableNum1.Text[0];
                    if(!IsValidHex(Nibble)) Nibble = '0';
                    NewQuant += Nibble;
                    Nibble = txtQuantizerTableNum1.Text[1];
                    if(!IsValidHex(Nibble)) Nibble = '0';
                    NewQuant += Nibble;

                    // Read out the content of the TextBox and
                    // check to get only the valid HEX value chars
                    for(int x = 0; x < txtQuantizer1.Text.Length; x++)
                    {
                      Nibble = txtQuantizer1.Text[x];
                      if(IsValidHex(Nibble))
                        NewQuant += Nibble.ToString();
                    }

                    // Check to make sure the size of the new
                    // huffman table is correct and if not, fix
                    if((NewQuant.Length % 2) == 1)
                      NewQuant += "0";

                    // Recalculated the size of the field and
                    // write back to the new file string
                    // for the 2 bytes of size
                    t = (NewQuant.Length + 4)/2;
                    Byte2 = (byte)(t % 256);
                    t >>= 8;
                    Byte1 = (byte)(t % 256);
                    NewData[SizeIndex] = Byte1;
                    SizeIndex++;
                    NewData[SizeIndex] = Byte2;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // Now write the new huffman table to the
                    // NewData string.
                    for(int x = 0; x < NewQuant.Length; x+=2)
                    {
                      NewData[count] = SetByteValue(
                        NewQuant[x], NewQuant[x+1]);
                      count++;
                      Loading.UpdateAndIncrement();
                      this.Update();
                    }
                  }
                  else if(QuantizerNumber == 1)
                  {
                    // Update the table we're reading
                    QuantizerNumber++;

                    // Get the table number
                    if(txtQuantizerTableNum2.Text.Length < 2)
                      txtQuantizerTableNum2.Text = "0" + "1";
                    Nibble = txtQuantizerTableNum2.Text[0];
                    if(!IsValidHex(Nibble)) Nibble = '0';
                    NewQuant += Nibble;
                    Nibble = txtQuantizerTableNum2.Text[1];
                    if(!IsValidHex(Nibble)) Nibble = '0';
                    NewQuant += Nibble;

                    // Read out the content of the TextBox and
                    // check to get only the valid HEX value chars
```



```
                    for(int x = 0; x < txtQuantizer2.Text.Length; x++)
                    {
                      Nibble = txtQuantizer2.Text[x];
                      if(IsValidHex(Nibble))
                        NewQuant += Nibble.ToString();
                    }

                    // Check to make sure the size of the new
                    // huffman table is correct and if not, fix
                    if((NewQuant.Length % 2) == 1)
                      NewQuant += "0";

                    // Recalculated the size of the field and
                    // write back to the new file string
                    // for the 2 bytes of size
                    t = (NewQuant.Length + 4)/2;
                    Byte2 = (byte)(t % 256);
                    t >>= 8;
                    Byte1 = (byte)(t % 256);
                    NewData[SizeIndex] = Byte1;
                    SizeIndex++;
                    NewData[SizeIndex] = Byte2;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // Now write the new huffman table to the
                    // NewData string.
                    for(int x = 0; x < NewQuant.Length; x+=2)
                    {
                      NewData[count] = SetByteValue(
                        NewQuant[x], NewQuant[x+1]);
                      count++;
                      Loading.UpdateAndIncrement();
                      this.Update();
                    }
                  }
                  else if(QuantizerNumber == 2)
                  {
                    // Update the table we're reading
                    QuantizerNumber++;

                    // Get the table number
                    if(txtQuantizerTableNum3.Text.Length < 2)
                      txtQuantizerTableNum3.Text = "0" + "2";
                    Nibble = txtQuantizerTableNum3.Text[0];
                    if(!IsValidHex(Nibble)) Nibble = '0';
                    NewQuant += Nibble;
                    Nibble = txtQuantizerTableNum3.Text[1];
                    if(!IsValidHex(Nibble)) Nibble = '0';
                    NewQuant += Nibble;

                    // Read out the content of the TextBox and
                    // check to get only the valid HEX value chars
                    for(int x = 0; x < txtQuantizer3.Text.Length; x++)
                    {
                      Nibble = txtQuantizer3.Text[x];
                      if(IsValidHex(Nibble))
                        NewQuant += Nibble.ToString();
                    }

                    // Check to make sure the size of the new
                    // huffman table is correct and if not, fix
                    if((NewQuant.Length % 2) == 1)
                      NewQuant += "0";

                    // Recalculated the size of the field and
```

```
                    // write back to the new file string
                    // for the 2 bytes of size
                    t = (NewQuant.Length + 4)/2;
                    Byte2 = (byte)(t % 256);
                    t >>= 8;
                    Byte1 = (byte)(t % 256);
                    NewData[SizeIndex] = Byte1;
                    SizeIndex++;
                    NewData[SizeIndex] = Byte2;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    Loading.UpdateAndIncrement();
                    this.Update();

                    // Now write the new huffman table to the
                    // NewData string.
                    for(int x = 0; x < NewQuant.Length; x+=2)
                    {
                      NewData[count] = SetByteValue(
                        NewQuant[x], NewQuant[x+1]);
                      count++;
                      Loading.UpdateAndIncrement();
                      this.Update();
                    }
                  }
                  else if(QuantizerNumber == 3)
                  {
                    // Update the table we're reading
                    QuantizerNumber++;

                    // Get the table number
                    if(txtQuantizerTableNum4.Text.Length < 2)
                      txtQuantizerTableNum4.Text = "0" + "3";
                    Nibble = txtQuantizerTableNum4.Text[0];
                    if(!IsValidHex(Nibble)) Nibble = '0';
                    NewQuant += Nibble;
                    Nibble = txtQuantizerTableNum4.Text[1];
                    if(!IsValidHex(Nibble)) Nibble = '0';
                    NewQuant += Nibble;

                    // Read out the content of the TextBox and
                    // check to get only the valid HEX value chars
                    for(int x = 0; x < txtQuantizer4.Text.Length; x++)
                    {
                      Nibble = txtQuantizer4.Text[x];
                      if(IsValidHex(Nibble))
                        NewQuant += Nibble.ToString();
                    }

                    // Check to make sure the size of the new
                    // huffman table is correct and if not, fix
                    if((NewQuant.Length % 2) == 1)
                      NewQuant += "0";

                    // Recalculated the size of the field and
                    // write back to the new file string
                    // for the 2 bytes of size
                    t = (NewQuant.Length + 4)/2;
                    Byte2 = (byte)(t % 256);
                    t >>= 8;
                    Byte1 = (byte)(t % 256);
                    NewData[SizeIndex] = Byte1;
                    SizeIndex++;
                    NewData[SizeIndex] = Byte2;

                    // Update the loading form
                    Loading.UpdateAndIncrement();
                    Loading.UpdateAndIncrement();
```



```
                    this.Update();

                    // Now write the new huffman table to the
                    // NewData string.
                    for(int x = 0; x < NewQuant.Length; x+=2)
                    {
                      NewData[count] = SetByteValue(
                        NewQuant[x], NewQuant[x+1]);
                      count++;
                      Loading.UpdateAndIncrement();
                      this.Update();
                    }
                  }
                  else
                  {
                    // Output an error
                    txtError.Text +=
                      "\nError: Too Many Quantizer Tables!! " +
                      "\n\t-- Marker ff" + C.ToString() + D.ToString()+
                      " was found in the original stream. " +
                                              "Marker and data NOT written to
new file.";
                    txtError.Update();
                    return false;
                  }

                  break;
                }

                case 'c':
                {  // Marker ffdc : Define number of lines, 4 bytes

                  byte Byte1;
                  byte Byte2;
                  byte Byte3;
                  byte Byte4;
                  int t;

                  t = NumberOfLines;

                                        Byte4 = (byte)(t % 256);
                  t >>= 8;
                  Byte3 = (byte)(t % 256);
                  t >>= 8;
                  Byte2 = (byte)(t % 256);
                  t >>= 8;
                  Byte1 = (byte)(t % 256);

                  NewData[count] = Byte1;
                  count++;
                  Loading.UpdateAndIncrement();
                  NewData[count] = Byte2;
                  count++;
                  Loading.UpdateAndIncrement();
                  NewData[count] = Byte3;
                  count++;
                  Loading.UpdateAndIncrement();
                  NewData[count] = Byte4;
                  count++;
                  Loading.UpdateAndIncrement();

                  this.Update();

                  break;
                }

                case 'd':
                {  // Marker ffdd : Define restart interval, 4 bytes
```

```
            byte Byte1;
            byte Byte2;
            byte Byte3;
            byte Byte4;
            int t;

            t = RestartInterval;

            Byte4 = (byte)(t % 256);
            t >>= 8;
            Byte3 = (byte)(t % 256);
            t >>= 8;
            Byte2 = (byte)(t % 256);
            t >>= 8;
            Byte1 = (byte)(t % 256);

            NewData[count] = Byte1;
            count++;
            Loading.UpdateAndIncrement();
            NewData[count] = Byte2;
            count++;
            Loading.UpdateAndIncrement();
            NewData[count] = Byte3;
            count++;
            Loading.UpdateAndIncrement();
            NewData[count] = Byte4;
            count++;
            Loading.UpdateAndIncrement();

            this.Update();

            break;
          }

          case 'e':
          {  // Marker ffde : Define Hierarchial Progression

            byte Byte1;
            byte Byte2;
            int SizeIndex = count;
            int t;
            string Progression = "";
            char Nibble;

            // Check to see if loading canceled
            if(Loading.Canceled)
            {
              Loading.Dispose();
              return false;
            }

            // Move past the size field
            count++;
            count++;

            // Read out the contents of the interface
            for(int x = 0; x < txtHierarchial.Text.Length; x++)
            {
              Nibble = txtHierarchial.Text[x];
              if(IsValidHex(Nibble))
                Progression += Nibble.ToString();
            }

            // Check the size of the new field
            if((Progression.Length % 2) == 1)
              Progression += "0";

            // Calculate the new size
            t = ((Progression.Length + 4)/2);
```



```
            Byte2 = (byte)(t % 256);
            t >>= 8;
            Byte1 = (byte)(t % 256);
            NewData[SizeIndex] = Byte1;
            SizeIndex++;
            NewData[SizeIndex] = Byte2;

            // Update the loading form
            Loading.UpdateAndIncrement();
            Loading.UpdateAndIncrement();
            this.Update();

            // Write the new values to NewData
            for(int x = 0; x < Progression.Length; x+=2)
            {
              NewData[count] = SetByteValue(
                Progression[x], Progression[x+1]);
              count++;

              Loading.UpdateAndIncrement();
              this.Update();
            }

            // Check to see if loading canceled
            if(Loading.Canceled)
            {
              Loading.Dispose();
              return false;
            }

            break;
          }

          case 'f':
          {  // Marker ffdf : Expand Reference Images, 3 bytes

            // Read out 3 bytes
            byte Byte1;
            byte Byte2;
            byte Byte3;
            int t;

            t = ExpandImage;

            Byte3 = (byte)(t % 256);
            t >>= 8;
            Byte2 = (byte)(t % 256);
            t >>= 8;
            Byte1 = (byte)(t % 256);

            NewData[count] = Byte1;
            count++;
            Loading.UpdateAndIncrement();
            NewData[count] = Byte2;
            count++;
            Loading.UpdateAndIncrement();
            NewData[count] = Byte3;
            count++;
            Loading.UpdateAndIncrement();

            this.Update();

            break;
          }

          default:
          {
            txtError.Text +=
              "\nError: Invalid File Marker Read!! " +
```

```
                    "\n\t-- Marker ffd" + D.ToString()+
                    " was found in the original file stream. " +
                    "Marker and data not written to the new file.";
                  txtError.Update();
                  break;
                }

              }  // End of: switch(D)

              break;

            } // End of: case 'd': // marker ffdX


            case 'e': // marker ffeX
            {      // e0 to ef - Reserved for application data
              byte Byte1;
              byte Byte2;
              int SizeIndex = count;
              int t;
              string NewAppData = "";
              char Nibble;

              // Check to see if loading canceled
              if(Loading.Canceled)
              {
                Loading.Dispose();
                return false;
              }

              // Move past size field
              count++;
              count++;

              // Get the correct table
              if(AppDataNumber == 0)
              {
                AppDataNumber++;

                // Read out the interface data
                for(int x = 0; x < txtApplicationData1.Text.Length; x++)
                {
                  Nibble = txtApplicationData1.Text[x];
                  if(IsValidHex(Nibble))
                    NewAppData += Nibble.ToString();
                }

                // Check the size of the new data
                if((NewAppData.Length % 2) == 1)
                  NewAppData += "0";

                // Calculate the size field
                t = ((NewAppData.Length + 4)/2);
                Byte2 = (byte)(t % 256);
                t >>= 8;
                Byte1 = (byte)(t % 256);
                NewData[SizeIndex] = Byte1;
                SizeIndex++;
                NewData[SizeIndex] = Byte2;

                // Update the loading form
                Loading.UpdateAndIncrement();
                Loading.UpdateAndIncrement();
                this.Update();

                // Write the new values to NewData
                for(int x = 0; x < NewAppData.Length; x+=2)
                {
                  NewData[count] = SetByteValue(
```



```
                    NewAppData[x], NewAppData[x+1]);
                  count++;

                  Loading.UpdateAndIncrement();
                  this.Update();
                }
              }
              else if(AppDataNumber == 1)
              {
                AppDataNumber++;

                // Read out the interface data
                for(int x = 0; x < txtApplicationData2.Text.Length; x++)
                {
                  Nibble = txtApplicationData2.Text[x];
                  if(IsValidHex(Nibble))
                    NewAppData += Nibble.ToString();
                }

                // Check the size of the new data
                if((NewAppData.Length % 2) == 1)
                  NewAppData += "0";

                // Calculate the size field
                t = ((NewAppData.Length + 4)/2);
                Byte2 = (byte)(t % 256);
                t >>= 8;
                Byte1 = (byte)(t % 256);
                NewData[SizeIndex] = Byte1;
                SizeIndex++;
                NewData[SizeIndex] = Byte2;

                // Update the loading form
                Loading.UpdateAndIncrement();
                Loading.UpdateAndIncrement();
                this.Update();

                // Write the new values to NewData
                for(int x = 0; x < NewAppData.Length; x+=2)
                {
                  NewData[count] = SetByteValue(
                    NewAppData[x], NewAppData[x+1]);
                  count++;

                  Loading.UpdateAndIncrement();
                  this.Update();
                }
              }
              else if(AppDataNumber == 2)
              {
                AppDataNumber++;

                // Read out the interface data
                for(int x = 0; x < txtApplicationData3.Text.Length; x++)
                {
                  Nibble = txtApplicationData3.Text[x];
                  if(IsValidHex(Nibble))
                    NewAppData += Nibble.ToString();
                }

                // Check the size of the new data
                if((NewAppData.Length % 2) == 1)
                  NewAppData += "0";

                // Calculate the size field
                t = ((NewAppData.Length + 4)/2);
                Byte2 = (byte)(t % 256);
                t >>= 8;
                Byte1 = (byte)(t % 256);
```

```
            NewData[SizeIndex] = Byte1;
            SizeIndex++;
            NewData[SizeIndex] = Byte2;

            // Update the loading form
            Loading.UpdateAndIncrement();
            Loading.UpdateAndIncrement();
            this.Update();

            // Write the new values to NewData
            for(int x = 0; x < NewAppData.Length; x+=2)
            {
              NewData[count] = SetByteValue(
                NewAppData[x], NewAppData[x+1]);
              count++;

              Loading.UpdateAndIncrement();
              this.Update();
            }
          }
          else if(AppDataNumber == 3)
          {
            AppDataNumber++;

            // Read out the interface data
            for(int x = 0; x < txtApplicationData4.Text.Length; x++)
            {
              Nibble = txtApplicationData4.Text[x];
              if(IsValidHex(Nibble))
                NewAppData += Nibble.ToString();
            }

            // Check the size of the new data
            if((NewAppData.Length % 2) == 1)
              NewAppData += "0";

            // Calculate the size field
            t = ((NewAppData.Length + 4)/2);
            Byte2 = (byte)(t % 256);
            t >>= 8;
            Byte1 = (byte)(t % 256);
            NewData[SizeIndex] = Byte1;
            SizeIndex++;
            NewData[SizeIndex] = Byte2;

            // Update the loading form
            Loading.UpdateAndIncrement();
            Loading.UpdateAndIncrement();
            this.Update();

            // Write the new values to NewData
            for(int x = 0; x < NewAppData.Length; x+=2)
            {
              NewData[count] = SetByteValue(
                NewAppData[x], NewAppData[x+1]);
              count++;

              Loading.UpdateAndIncrement();
              this.Update();
            }
          }
          else if(AppDataNumber == 4)
          {
            AppDataNumber++;

            // Read out the interface data
            for(int x = 0; x < txtApplicationData5.Text.Length; x++)
            {
              Nibble = txtApplicationData5.Text[x];
```



```
              if(IsValidHex(Nibble))
                NewAppData += Nibble.ToString();
            }

            // Check the size of the new data
            if((NewAppData.Length % 2) == 1)
              NewAppData += "0";

            // Calculate the size field
            t = ((NewAppData.Length + 4)/2);
            Byte2 = (byte)(t % 256);
            t >>= 8;
            Byte1 = (byte)(t % 256);
            NewData[SizeIndex] = Byte1;
            SizeIndex++;
            NewData[SizeIndex] = Byte2;

            // Update the loading form
            Loading.UpdateAndIncrement();
            Loading.UpdateAndIncrement();
            this.Update();

            // Write the new values to NewData
            for(int x = 0; x < NewAppData.Length; x+=2)
            {
              NewData[count] = SetByteValue(
                NewAppData[x], NewAppData[x+1]);
              count++;

              Loading.UpdateAndIncrement();
              this.Update();
            }
          }
          else if(AppDataNumber == 5)
          {
            AppDataNumber++;

            // Read out the interface data
            for(int x = 0; x < txtApplicationData6.Text.Length; x++)
            {
              Nibble = txtApplicationData6.Text[x];
              if(IsValidHex(Nibble))
                NewAppData += Nibble.ToString();
            }

            // Check the size of the new data
            if((NewAppData.Length % 2) == 1)
              NewAppData += "0";

            // Calculate the size field
            t = ((NewAppData.Length + 4)/2);
            Byte2 = (byte)(t % 256);
            t >>= 8;
            Byte1 = (byte)(t % 256);
            NewData[SizeIndex] = Byte1;
            SizeIndex++;
            NewData[SizeIndex] = Byte2;

            // Update the loading form
            Loading.UpdateAndIncrement();
            Loading.UpdateAndIncrement();
            this.Update();

            // Write the new values to NewData
            for(int x = 0; x < NewAppData.Length; x+=2)
            {
              NewData[count] = SetByteValue(
                NewAppData[x], NewAppData[x+1]);
              count++;
```

```
        Loading.UpdateAndIncrement();
        this.Update();
      }
    }
    else if(AppDataNumber == 6)
    {
      AppDataNumber++;

      // Read out the interface data
      for(int x = 0; x < txtApplicationData7.Text.Length; x++)
      {
        Nibble = txtApplicationData7.Text[x];
        if(IsValidHex(Nibble))
          NewAppData += Nibble.ToString();
      }

      // Check the size of the new data
      if((NewAppData.Length % 2) == 1)
        NewAppData += "0";

      // Calculate the size field
      t = ((NewAppData.Length + 4)/2);
      Byte2 = (byte)(t % 256);
      t >>= 8;
      Byte1 = (byte)(t % 256);
      NewData[SizeIndex] = Byte1;
      SizeIndex++;
      NewData[SizeIndex] = Byte2;

      // Update the loading form
      Loading.UpdateAndIncrement();
      Loading.UpdateAndIncrement();
      this.Update();

      // Write the new values to NewData
      for(int x = 0; x < NewAppData.Length; x+=2)
      {
        NewData[count] = SetByteValue(
          NewAppData[x], NewAppData[x+1]);
        count++;

        Loading.UpdateAndIncrement();
        this.Update();
      }
    }
    else if(AppDataNumber == 7)
    {
      AppDataNumber++;

      // Read out the interface data
      for(int x = 0; x < txtApplicationData8.Text.Length; x++)
      {
        Nibble = txtApplicationData8.Text[x];
        if(IsValidHex(Nibble))
          NewAppData += Nibble.ToString();
      }

      // Check the size of the new data
      if((NewAppData.Length % 2) == 1)
        NewAppData += "0";

      // Calculate the size field
      t = ((NewAppData.Length + 4)/2);
      Byte2 = (byte)(t % 256);
      t >>= 8;
      Byte1 = (byte)(t % 256);
      NewData[SizeIndex] = Byte1;
      SizeIndex++;
```



```
      NewData[SizeIndex] = Byte2;

      // Update the loading form
      Loading.UpdateAndIncrement();
      Loading.UpdateAndIncrement();
      this.Update();

      // Write the new values to NewData
      for(int x = 0; x < NewAppData.Length; x+=2)
      {
        NewData[count] = SetByteValue(
          NewAppData[x], NewAppData[x+1]);
        count++;

        Loading.UpdateAndIncrement();
        this.Update();
      }
    }
    else if(AppDataNumber == 8)
    {
      AppDataNumber++;

      // Read out the interface data
      for(int x = 0; x < txtApplicationData9.Text.Length; x++)
      {
        Nibble = txtApplicationData9.Text[x];
        if(IsValidHex(Nibble))
          NewAppData += Nibble.ToString();
      }

      // Check the size of the new data
      if((NewAppData.Length % 2) == 1)
        NewAppData += "0";

      // Calculate the size field
      t = ((NewAppData.Length + 4)/2);
      Byte2 = (byte)(t % 256);
      t >>= 8;
      Byte1 = (byte)(t % 256);
      NewData[SizeIndex] = Byte1;
      SizeIndex++;
      NewData[SizeIndex] = Byte2;

      // Update the loading form
      Loading.UpdateAndIncrement();
      Loading.UpdateAndIncrement();
      this.Update();

      // Write the new values to NewData
      for(int x = 0; x < NewAppData.Length; x+=2)
      {
        NewData[count] = SetByteValue(
          NewAppData[x], NewAppData[x+1]);
        count++;

        Loading.UpdateAndIncrement();
        this.Update();
      }
    }
    else if(AppDataNumber == 9)
    {
      AppDataNumber++;

      // Read out the interface data
      for(int x = 0; x < txtApplicationData10.Text.Length; x++)
      {
        Nibble = txtApplicationData10.Text[x];
        if(IsValidHex(Nibble))
          NewAppData += Nibble.ToString();
```

```
                }

                // Check the size of the new data
                if((NewAppData.Length % 2) == 1)
                  NewAppData += "0";

                // Calculate the size field
                t = ((NewAppData.Length + 4)/2);
                Byte2 = (byte)(t % 256);
                t >>= 8;
                Byte1 = (byte)(t % 256);
                NewData[SizeIndex] = Byte1;
                SizeIndex++;
                NewData[SizeIndex] = Byte2;

                // Update the loading form
                Loading.UpdateAndIncrement();
                Loading.UpdateAndIncrement();
                this.Update();

                // Write the new values to NewData
                for(int x = 0; x < NewAppData.Length; x+=2)
                {
                  NewData[count] = SetByteValue(
                    NewAppData[x], NewAppData[x+1]);
                  count++;

                  Loading.UpdateAndIncrement();
                  this.Update();
                }
              }
              else
              {
                // Output an error
                txtError.Text +=
                  "\nError: Too Many Application Data frames!! " +
                  "\n\t-- Marker ff" + C.ToString() + D.ToString()+
                  " was found in the original stream. "+
                                  "Marker and data NOT written to new file
."";

                txtError.Update();
              }

              break;
            }

            case 'f': // marker fffX
            {
              switch(D)
              {
                case '0': goto case 'd';
                case '1': goto case 'd';
                case '2': goto case 'd';
                case '3': goto case 'd';
                case '4': goto case 'd';
                case '5': goto case 'd';
                case '6': goto case 'd';
                case '7': goto case 'd';
                case '8': goto case 'd';
                case '9': goto case 'd';
                case 'a': goto case 'd';
                case 'b': goto case 'd';
                case 'c': goto case 'd';
                case 'd':
                {  // marker fff0 to fffd: Reserved for JPEG extensions

                  txtError.Text +=
                    "\nError: Reserved ofr JPEG Extensions marker found!!"+
                    "\n\t-- Marker ff" + C.ToString() + D.ToString()+
```



```
                    " was found in the original stream. "+
                                    "Marker and data NOT written to new fi
le.";
                  txtError.Update();
                  break;
                }

                case 'e': // marker fffe - Comments
                {
                  byte Byte1;
                  byte Byte2;
                  int SizeIndex = count;
                  int t;
                  string NewComments = "";
                  char Nibble;

                  // Check if loading canceled
                  if(Loading.Canceled)
                  {
                    Loading.Dispose();
                    return false;
                  }

                  // Read out the interface data
                  for(int x = 0; x < txtComments.Text.Length; x++)
                  {
                    Nibble = txtComments.Text[x];
                    NewComments += Nibble.ToString();
                  }

                  // Calculate the new field size
                  t = NewComments.Length + 2;
                  Byte2 = (byte)(t % 256);
                  t >>= 8;
                  Byte1 = (byte)(t % 256);
                  NewData[SizeIndex] = Byte1;
                  SizeIndex++;
                  NewData[SizeIndex] = Byte2;

                  // Update the loading form
                  Loading.UpdateAndIncrement();
                  Loading.UpdateAndIncrement();
                  this.Update();

                  // Write the new vales to NewData
                  for(int x = 0; x < NewComments.Length; x++)
                  {
                    NewData[count] = (byte)NewComments[x];
                    count++;

                    Loading.UpdateAndIncrement();
                    this.Update();
                  }
                  break;
                }
                case 'f': // marker ffff -- Marker Not Defined
                {
                  txtError.Text +=
                    "\nError: Marker NOT defined " +
                    "\n\t-- Marker ffff was found in the original file "+
                                    "stream.\nMarker and Data not written
to the new file.";
                  txtError.Update();
                  break;
                }

                default:
                {
                  txtError.Text +=
```

```
                    "\nError: Invalid File Marker Read!! " +
                    "\n\t-- Marker ffd" + D.ToString()+
                    " was found in the original file stream. " +
                    "Marker and Data not written to the new file.";
                  txtError.Update();
                  break;
                }

              } // End of: switch(D)

              break;
            }

            default:
            {
              txtError.Text +=
                "\nError: Invalid File Marker Read!! " +
                "\n\t-- Marker ff" + C.ToString() + D.ToString()+
                " was found in the original file stream. " +
                "Marker and Data not written to the new file.";
              txtError.Update();
              break;
            }

          }  // End of: switch(Top1)

        }  // End of: if(Top1 == 'f' && Bottom1 == 'f')
        else
        {
          if(ShowWarning(
            "\nYou have an invalid marker!"))
          {
            txtError.Text +=
              "\nError: Invalid Marker Found!! " +
              "\n\t-- Marker ff" + C.ToString() + D.ToString() +
              " was found in the original file stream. " +
              "Marker and Data not written to the new file.";
            txtError.Update();
            ShowWarning(
              "\nYou have an invalid marker! Do you want to continue "+
                           "to write to file?");
            break;
          }

        }
      } // End of: while(A != 'f' && B != 'f' && C != 'd' && D != 'a')

    }
    catch(Exception ex)
    {
      if(!ShowWarning(
        "Warning, an exception occured:\n\n" +
        "Exception Error:\n" +
        ex.Message + "\n\nWas throw by:\n" +
        ex.Source +
        "\n\nNot all write operations completed for this updated file,"+
              " do you want to continue with the load operation?" +
        "\n(if you choose to continue you will have data loss)",
        "Load File Exception"))
      {
        Loading.Dispose();
        return false;
      }
      ClearInterfaceData();
    }

    Loading.Dispose();
    return true;
```



```
  }  // End of: private void CreatedManipulatedPicture()


  /// <summary>
  /// Pre-conditions:    None.
  ///  Post-conditions:
  ///    All of the data for the new JPEG image being created is written to
  ///    the file name contained in the txtManipulatedFile TextBox field.
  ///  Description:
  ///    The purpose of this method is to create a new manipulated image
  ///    based upon all of the data currently loaded within the Manipulator.
  ///    To perform this functionality, this function should call the
  ///    CreateManipulatedPicture() method to create a file string to store
  ///    the new file data.  Then, this function should call the WriteFile()
  ///    method to write all of this data to the new file.  Then, to update
  ///    the Manipulated picture files, this function should call the
  ///    UpdateManipulatedPicture() method.  Lastly, this method should do
  ///    some error checking to make sure this function executes properly.
  ///    If an error is encountered, then the ShowWarning() method should
  ///    be called to display the error to the user and the txtError
  ///    TextBox control should be updated with this error information.
  /// </summary>
  private void CreateISEImage()
  {
    if(!LoadingInterface)
    {
      if(CreateManipulatedPicture(ref NewData))
      {
        if(ISE != null)
        {
          ISE.Dispose();
          ISE = null;
          ISEsmall.Dispose();
          ISEsmall = null;
        }
        WriteFile(ref NewData);
        UpdateManipulatedPicture(this.txtManipulatedFile.Text.Trim());
      }
    }
    else
    {
      ShowWarning(
        "The interface is STILL being loaded, you cannot create a " +
        "new file until load has finished.",
        "Cannot Create New File!");
    }
  }

  #endregion Methods to Convert from ACSII to Binary


  #endregion ISE Coded Functions

  #region Created by Windows Form Designer

  //
  // Variables created by the Visual Studio .NET Form Designer
  //
  private System.Windows.Forms.MainMenu menuFrmMain;
  private System.Windows.Forms.MenuItem menuFile;

  private System.Windows.Forms.PictureBox picOriginal;
  private System.Windows.Forms.PictureBox picManipulated;
  private System.Windows.Forms.PictureBox picOriginalSmall;
  private System.Windows.Forms.PictureBox picManipulatedSmall;

  private System.Windows.Forms.MenuItem menuOpen;
  private System.Windows.Forms.MenuItem menuExit;
```

```
    private System.Windows.Forms.OpenFileDialog openFileDialog;

    private System.ComponentModel.IContainer components;

    private System.Windows.Forms.ToolTip toolTips;

    private System.Windows.Forms.TabControl tabMain;

    private System.Windows.Forms.TabPage tabConsol;
    private System.Windows.Forms.TabPage tabOriginal;
    private System.Windows.Forms.TabPage tabManipulated;
    private System.Windows.Forms.SaveFileDialog saveFileDialog1;
    private System.Windows.Forms.OpenFileDialog openFileDialog1;
    private System.Windows.Forms.MenuItem menuOpenProject;
    private System.Windows.Forms.MenuItem menuSaveProject;
    private System.Windows.Forms.MenuItem menuItem1;
    private System.Windows.Forms.MenuItem menuNewProject;
    private System.Windows.Forms.MenuItem menuItem3;
    private System.Windows.Forms.MenuItem menuEdit;
    private System.Windows.Forms.MenuItem menuCopy;
    private System.Windows.Forms.MenuItem menuCut;
    private System.Windows.Forms.MenuItem menuPaste;
    private System.Windows.Forms.MenuItem menuUpdate;
    private System.Windows.Forms.MenuItem menuView;
    private System.Windows.Forms.MenuItem menuStretchMode;
    private System.Windows.Forms.MenuItem menuSmallOriginal;
    private System.Windows.Forms.MenuItem menuLargeOriginal;
    private System.Windows.Forms.MenuItem menuLargeManipulated;
    private System.Windows.Forms.MenuItem menuSmallManipulated;
    private System.Windows.Forms.MenuItem menuAll;
    private System.Windows.Forms.TabPage tabProject;
    private System.Windows.Forms.Label lblNotes;
    private System.Windows.Forms.Button btnUpdatePicture;
    private System.Windows.Forms.Button btnSavePicture;
    private System.Windows.Forms.Button btnLoadPicture;
    private System.Windows.Forms.Label lblFilePath;
    private System.Windows.Forms.TextBox txtProjectPath;
    private System.Windows.Forms.Button btnLoad;
    private System.Windows.Forms.Button btnSave;
    private System.Windows.Forms.Button btnNew;
    private System.Windows.Forms.TextBox txtNotes;
    private System.Windows.Forms.TabPage tabFile;
    private System.Windows.Forms.Label lblComments;
    private System.Windows.Forms.TextBox txtComments;
    private System.Windows.Forms.TextBox txtFileSize;
    private System.Windows.Forms.Label lblFileSize;
    private System.Windows.Forms.Label lblManipulatedFile;
    private System.Windows.Forms.TextBox txtManipulatedFile;
    private System.Windows.Forms.Label lblOriginalFile;
    private System.Windows.Forms.TextBox txtOriginalFile;
    private System.Windows.Forms.TabPage tabHeaders;
    private System.Windows.Forms.Label lblComponents;
    private System.Windows.Forms.Label lblNumberImageComponents;
    private System.Windows.Forms.Label lblNumberHuffmanSamples;
    private System.Windows.Forms.Label lblNumberHuffmanLines;
    private System.Windows.Forms.Label lblPrecision;
    private System.Windows.Forms.Label lblStartHuffmanSize;
    private System.Windows.Forms.Label lblStartHuffman;
    private System.Windows.Forms.RichTextBox txtComponents;
    private System.Windows.Forms.TextBox txtNumberImageComponents;
    private System.Windows.Forms.TextBox txtNumberHuffmanSamples;
    private System.Windows.Forms.TextBox txtNumberHuffmanLines;
    private System.Windows.Forms.TextBox txtPrecision;
    private System.Windows.Forms.TextBox txtStartHuffmanSize;
    private System.Windows.Forms.TextBox txtStartHuffman;
    private System.Windows.Forms.TabPage tabHuffman1;
    private System.Windows.Forms.Button btnClearHuffman4;
    private System.Windows.Forms.Button btnAddRandomHuffman4;
    private System.Windows.Forms.Button btnClearHuffman2;
```



```
    private System.Windows.Forms.Button btnAddRandomHuffman2;
    private System.Windows.Forms.Button btnClearHuffman3;
    private System.Windows.Forms.Button btnAddRandomHuffman3;
    private System.Windows.Forms.Button btnClearHuffman1;
    private System.Windows.Forms.Button btnAddRandomHuffman1;
    private System.Windows.Forms.Button btnRestoreHuffman4;
    private System.Windows.Forms.Button btnRestoreHuffman3;
    private System.Windows.Forms.Button btnRestoreHuffman2;
    private System.Windows.Forms.Button btnRestoreHuffman1;
    private System.Windows.Forms.TextBox txtHuffmanOriginal4;
    private System.Windows.Forms.Label lblHuffmanOriginalMarker4;
    private System.Windows.Forms.Label lblHuffmanOriginal4;
    private System.Windows.Forms.TextBox txtHuffman4;
    private System.Windows.Forms.Label lblHuffmanMarker4;
    private System.Windows.Forms.Label lblHuffman4;
    private System.Windows.Forms.TextBox txtHuffmanOriginal2;
    private System.Windows.Forms.Label lblHuffmanOriginalMarker2;
    private System.Windows.Forms.Label lblHuffmanOriginal2;
    private System.Windows.Forms.TextBox txtHuffman2;
    private System.Windows.Forms.Label lblHuffmanMarker2;
    private System.Windows.Forms.Label lblHuffman2;
    private System.Windows.Forms.TextBox txtHuffmanOriginal3;
    private System.Windows.Forms.Label lblHuffmanOriginalMarker3;
    private System.Windows.Forms.Label lblHuffmanOriginal3;
    private System.Windows.Forms.TextBox txtHuffman3;
    private System.Windows.Forms.Label lblHuffmanMarker3;
    private System.Windows.Forms.Label lblHuffman3;
    private System.Windows.Forms.TextBox txtHuffmanOriginal1;
    private System.Windows.Forms.Label lblHuffmanOriginalMarker1;
    private System.Windows.Forms.Label lblHuffmanOriginal1;
    private System.Windows.Forms.TextBox txtHuffman1;
    private System.Windows.Forms.Label lblHuffmanMarker1;
    private System.Windows.Forms.Label lblHuffman1;
    private System.Windows.Forms.TabPage tabHuffman2;
    private System.Windows.Forms.Button btnClearHuffman8;
    private System.Windows.Forms.Button btnAddRandomHuffman8;
    private System.Windows.Forms.Button btnClearHuffman7;
    private System.Windows.Forms.Button btnAddRandomHuffman7;
    private System.Windows.Forms.Button btnClearHuffman6;
    private System.Windows.Forms.Button btnAddRandomHuffman6;
    private System.Windows.Forms.Button btnClearHuffman5;
    private System.Windows.Forms.Button btnAddRandomHuffman5;
    private System.Windows.Forms.Button btnRestoreHuffman8;
    private System.Windows.Forms.Button btnRestoreHuffman7;
    private System.Windows.Forms.Button btnRestoreHuffman6;
    private System.Windows.Forms.Button btnRestoreHuffman5;
    private System.Windows.Forms.TextBox txtHuffmanOriginal8;
    private System.Windows.Forms.Label lblHuffmanOriginalMarker8;
    private System.Windows.Forms.Label lblHuffmanOriginal8;
    private System.Windows.Forms.TextBox txtHuffman8;
    private System.Windows.Forms.Label lblHuffmanMarker8;
    private System.Windows.Forms.Label lblHuffman8;
    private System.Windows.Forms.TextBox txtHuffmanOriginal6;
    private System.Windows.Forms.Label lblHuffmanOriginalMarker6;
    private System.Windows.Forms.Label lblHuffmanOriginal6;
    private System.Windows.Forms.TextBox txtHuffman6;
    private System.Windows.Forms.Label lblHuffmanMarker6;
    private System.Windows.Forms.Label lblHuffman6;
    private System.Windows.Forms.TextBox txtHuffmanOriginal7;
    private System.Windows.Forms.Label lblHuffmanOriginalMarker7;
    private System.Windows.Forms.Label lblHuffmanOriginal7;
    private System.Windows.Forms.TextBox txtHuffman7;
    private System.Windows.Forms.Label lblHuffmanMarker7;
    private System.Windows.Forms.Label lblHuffman7;
    private System.Windows.Forms.TextBox txtHuffmanOriginal5;
    private System.Windows.Forms.Label lblHuffmanOriginalMarker5;
    private System.Windows.Forms.Label lblHuffmanOriginal5;
    private System.Windows.Forms.TextBox txtHuffman5;
    private System.Windows.Forms.Label lblHuffmanMarker5;
```

```
    private System.Windows.Forms.Label lblHuffman5;
    private System.Windows.Forms.TabPage tabQuantizer;
    private System.Windows.Forms.Button btnClearQuantizer4;
    private System.Windows.Forms.Button btnAddRandomQuantizer4;
    private System.Windows.Forms.Button btnClearQuantizer3;
    private System.Windows.Forms.Button btnAddRandomQuantizer3;
    private System.Windows.Forms.Button btnClearQuantizer2;
    private System.Windows.Forms.Button btnAddRandomQuantizer2;
    private System.Windows.Forms.Button btnClearQuantizer1;
    private System.Windows.Forms.Button btnAddRandomQuantizer1;
    private System.Windows.Forms.Button btnRestoreQuantizer4;
    private System.Windows.Forms.Button btnRestoreQuantizer3;
    private System.Windows.Forms.Button btnRestoreQuantizer2;
    private System.Windows.Forms.Button btnRestoreQuantizer1;
    private System.Windows.Forms.TextBox txtQuantizerOriginal4;
    private System.Windows.Forms.Label lblQuantizerOriginalMarker4;
    private System.Windows.Forms.Label lblQuantizerOriginal4;
    private System.Windows.Forms.TextBox txtQuantizer4;
    private System.Windows.Forms.Label lblQuantizerMarker4;
    private System.Windows.Forms.Label lblQuantizer4;
    private System.Windows.Forms.TextBox txtQuantizerOriginal2;
    private System.Windows.Forms.Label lblQuantizerOriginalMarker2;
    private System.Windows.Forms.Label lblQuantizerOriginal2;
    private System.Windows.Forms.TextBox txtQuantizer2;
    private System.Windows.Forms.Label lblQuantizerMarker2;
    private System.Windows.Forms.Label lblQuantizer2;
    private System.Windows.Forms.TextBox txtQuantizerOriginal3;
    private System.Windows.Forms.Label lblQuantizerOriginalMarker3;
    private System.Windows.Forms.Label lblQuantizerOriginal3;
    private System.Windows.Forms.TextBox txtQuantizer3;
    private System.Windows.Forms.Label lblQuantizerMarker3;
    private System.Windows.Forms.Label lblQuantizer3;
    private System.Windows.Forms.TextBox txtQuantizerOriginal1;
    private System.Windows.Forms.Label lblQuantizerOriginalMarker1;
    private System.Windows.Forms.Label lblQuantizerOriginal1;
    private System.Windows.Forms.TextBox txtQuantizer1;
    private System.Windows.Forms.Label lblQuantizerMarker1;
    private System.Windows.Forms.Label lblQuantizer1;
    private System.Windows.Forms.TabPage tabEncodedData;
    private System.Windows.Forms.Label lblOriginalHeader;
    private System.Windows.Forms.TextBox txtOriginalHeader;
    private System.Windows.Forms.Label lblScanHeader;
    private System.Windows.Forms.TextBox txtScanHeader;
    private System.Windows.Forms.TextBox txtOriginalEncodedData;
    private System.Windows.Forms.Label lblOriginalEncodedData;
    private System.Windows.Forms.TextBox txtEncodedData;
    private System.Windows.Forms.Label lblEncodedData;
    private System.Windows.Forms.TabPage tabApplicationData;
    private System.Windows.Forms.TextBox txtApplicationData10;
    private System.Windows.Forms.Label lblApplicationMarker10;
    private System.Windows.Forms.Label lblApplicationData10;
    private System.Windows.Forms.TextBox txtApplicationData9;
    private System.Windows.Forms.Label lblApplicationMarker9;
    private System.Windows.Forms.Label lblApplicationData9;
    private System.Windows.Forms.TextBox txtApplicationData8;
    private System.Windows.Forms.Label lblApplicationMarker8;
    private System.Windows.Forms.Label lblApplicationData8;
    private System.Windows.Forms.TextBox txtApplicationData7;
    private System.Windows.Forms.Label lblApplicationMarker7;
    private System.Windows.Forms.Label lblApplicationData7;
    private System.Windows.Forms.TextBox txtApplicationData6;
    private System.Windows.Forms.Label lblApplicationMarker6;
    private System.Windows.Forms.Label lblApplicationData6;
    private System.Windows.Forms.TextBox txtApplicationData5;
    private System.Windows.Forms.Label lblApplicationMarker5;
    private System.Windows.Forms.Label lblApplicationData5;
    private System.Windows.Forms.TextBox txtApplicationData4;
    private System.Windows.Forms.Label lblApplicationMarker4;
    private System.Windows.Forms.Label lblApplicationData4;
```



```
    private System.Windows.Forms.TextBox txtApplicationData3;
    private System.Windows.Forms.Label lblApplicationMarker3;
    private System.Windows.Forms.Label lblApplicationData3;
    private System.Windows.Forms.TextBox txtApplicationData2;
    private System.Windows.Forms.Label lblApplicationMarker2;
    private System.Windows.Forms.Label lblApplicationData2;
    private System.Windows.Forms.TextBox txtApplicationData1;
    private System.Windows.Forms.Label lblApplicationMarker1;
    private System.Windows.Forms.Label lblApplicationData1;
    private System.Windows.Forms.TabPage tabMisc;
    private System.Windows.Forms.Label lblExpandMarker;
    private System.Windows.Forms.TextBox txtExpand;
    private System.Windows.Forms.Label lblExpand;
    private System.Windows.Forms.TextBox txtHierarchial;
    private System.Windows.Forms.Label lblHierarchialMarker;
    private System.Windows.Forms.Label lblHierarchial;
    private System.Windows.Forms.TextBox txtRestartMod8;
    private System.Windows.Forms.Label lblRestartMod8;
    private System.Windows.Forms.TextBox txtError;
    private System.Windows.Forms.Label lblError;
    private System.Windows.Forms.Label lblNumberLinesMarker;
    private System.Windows.Forms.Label lblRestartMarker;
    private System.Windows.Forms.TextBox txtNumberLines;
    private System.Windows.Forms.Label lblNumberLines;
    private System.Windows.Forms.TextBox txtRestart;
    private System.Windows.Forms.Label lblRestart;
    private System.Windows.Forms.Label lblQuantizerTableNum1;
    private System.Windows.Forms.Label txtQuantizerTableNum1;
    private System.Windows.Forms.Label txtQuantizerTableNum2;
    private System.Windows.Forms.Label lblQuantizerTableNum2;
    private System.Windows.Forms.Label txtQuantizerTableNum3;
    private System.Windows.Forms.Label lblQuantizerTableNum3;
    private System.Windows.Forms.Label txtQuantizerTableNum4;
    private System.Windows.Forms.Label lblQuantizerTableNum4;
    private System.Windows.Forms.TabControl tabSubConsole;


    #endregion Created by Windows Form Designer

    #region Standard Windows Form Application Methods


    /// <summary>
    /// Pre-conditions:  None.
    /// Post-conditions:
    ///    The frmMain Form of the application has been constructed.
    ///  Parameters:  None.
    ///  Return values:
    ///    Form constructor, no return type.
    ///  Description:
    ///    This is the constructor for the frmMain Form of the application.
    ///    This function will call the InitializeComponent() method and the
    ///    ISEConstructor() to initialize the application.
    /// </summary>
    public frmMain()
    {
      InitializeComponent();
      ISEConstructor();
    }


    /// <summary>
    /// Pre-conditions:  None.
    /// Post-conditions:
    ///    All of the memory and resources used in the frmMain have been
    ///    released.
    ///  Parameters:
    ///    TRUE to release both managed and unmanaged resources and FALSE to
    ///    release only unmanaged resources.
```

```
/// Return values:
///    Function returns void.
/// Description:
///    This function is called when the application is when the current
///    instance of the Form is destroyed.  It is not required, but
///    implementation of this method is recommended for .NET objects
///    that require large amounts of data, to ensure that all memory
///    allocated for the Form is freed immediately when the Form is
///    destroyed.
/// </summary>
protected override void Dispose( bool disposing )
{
  if( disposing )
  {
    if (components != null)
    {
      components.Dispose();
    }
  }
  base.Dispose( disposing );
}


#region Windows Form Designer generated code

/// <summary>
/// Pre-conditions:   None.
/// Post-conditions:
///    All of the variables created by the Visual Studio .NET Form
///    Designer have been initialized.
///  Parameters:      None.
///  Return values:
///    Function returns void.
///  Description:
///    This function is required to be called by the FormM-^Rs constructor.

///    It initializes all of the variables and values set with the form
///    designer at the beginning of the program execution.
/// </summary>
private void InitializeComponent()
{
  this.components = new System.ComponentModel.Container();
  System.Resources.ResourceManager resources = new
            System.Resources.ResourceManager(typeof(frmMain));
  this.menuFrmMain = new System.Windows.Forms.MainMenu();
  this.menuFile = new System.Windows.Forms.MenuItem();
  this.menuOpen = new System.Windows.Forms.MenuItem();
  this.menuUpdate = new System.Windows.Forms.MenuItem();
  this.menuItem1 = new System.Windows.Forms.MenuItem();
  this.menuNewProject = new System.Windows.Forms.MenuItem();
  this.menuOpenProject = new System.Windows.Forms.MenuItem();
  this.menuSaveProject = new System.Windows.Forms.MenuItem();
  this.menuItem3 = new System.Windows.Forms.MenuItem();
  this.menuExit = new System.Windows.Forms.MenuItem();
  this.menuEdit = new System.Windows.Forms.MenuItem();
  this.menuCopy = new System.Windows.Forms.MenuItem();
  this.menuCut = new System.Windows.Forms.MenuItem();
  this.menuPaste = new System.Windows.Forms.MenuItem();
  this.menuView = new System.Windows.Forms.MenuItem();
  this.menuStretchMode = new System.Windows.Forms.MenuItem();
  this.menuLargeOriginal = new System.Windows.Forms.MenuItem();
  this.menuLargeManipulated = new System.Windows.Forms.MenuItem();
  this.menuSmallOriginal = new System.Windows.Forms.MenuItem();
  this.menuSmallManipulated = new System.Windows.Forms.MenuItem();
  this.menuAll = new System.Windows.Forms.MenuItem();
  this.menuItem2 = new System.Windows.Forms.MenuItem();
  this.menuTutorial = new System.Windows.Forms.MenuItem();
  this.menuManual = new System.Windows.Forms.MenuItem();
```



```
  this.menuItem6 = new System.Windows.Forms.MenuItem();
  this.menuAbout = new System.Windows.Forms.MenuItem();
  this.tabMain = new System.Windows.Forms.TabControl();
  this.tabConsol = new System.Windows.Forms.TabPage();
  this.tabSubConsole = new System.Windows.Forms.TabControl();
  this.tabProject = new System.Windows.Forms.TabPage();
  this.lblNotes = new System.Windows.Forms.Label();
  this.btnUpdatePicture = new System.Windows.Forms.Button();
  this.btnSavePicture = new System.Windows.Forms.Button();
  this.btnLoadPicture = new System.Windows.Forms.Button();
  this.lblFilePath = new System.Windows.Forms.Label();
  this.txtProjectPath = new System.Windows.Forms.TextBox();
  this.btnLoad = new System.Windows.Forms.Button();
  this.btnSave = new System.Windows.Forms.Button();
  this.btnNew = new System.Windows.Forms.Button();
  this.txtNotes = new System.Windows.Forms.TextBox();
  this.tabFile = new System.Windows.Forms.TabPage();
  this.txtManipulatedFile = new System.Windows.Forms.TextBox();
  this.lblComments = new System.Windows.Forms.Label();
  this.txtComments = new System.Windows.Forms.TextBox();
  this.txtFileSize = new System.Windows.Forms.TextBox();
  this.lblFileSize = new System.Windows.Forms.Label();
  this.lblManipulatedFile = new System.Windows.Forms.Label();
  this.lblOriginalFile = new System.Windows.Forms.Label();
  this.txtOriginalFile = new System.Windows.Forms.TextBox();
  this.tabHeaders = new System.Windows.Forms.TabPage();
  this.lblComponents = new System.Windows.Forms.Label();
  this.lblNumberImageComponents = new System.Windows.Forms.Label();
  this.lblNumberHuffmanSamples = new System.Windows.Forms.Label();
  this.lblNumberHuffmanLines = new System.Windows.Forms.Label();
  this.lblPrecision = new System.Windows.Forms.Label();
  this.lblStartHuffmanSize = new System.Windows.Forms.Label();
  this.lblStartHuffman = new System.Windows.Forms.Label();
  this.txtComponents = new System.Windows.Forms.RichTextBox();
  this.txtNumberImageComponents = new System.Windows.Forms.TextBox();
  this.txtNumberHuffmanSamples = new System.Windows.Forms.TextBox();
  this.txtNumberHuffmanLines = new System.Windows.Forms.TextBox();
  this.txtPrecision = new System.Windows.Forms.TextBox();
  this.txtStartHuffmanSize = new System.Windows.Forms.TextBox();
  this.txtStartHuffman = new System.Windows.Forms.TextBox();
  this.tabHuffman1 = new System.Windows.Forms.TabPage();
  this.btnClearHuffman4 = new System.Windows.Forms.Button();
  this.btnAddRandomHuffman4 = new System.Windows.Forms.Button();
  this.btnClearHuffman2 = new System.Windows.Forms.Button();
  this.btnAddRandomHuffman2 = new System.Windows.Forms.Button();
  this.btnClearHuffman3 = new System.Windows.Forms.Button();
  this.btnAddRandomHuffman3 = new System.Windows.Forms.Button();
  this.btnClearHuffman1 = new System.Windows.Forms.Button();
  this.btnAddRandomHuffman1 = new System.Windows.Forms.Button();
  this.btnRestoreHuffman4 = new System.Windows.Forms.Button();
  this.btnRestoreHuffman3 = new System.Windows.Forms.Button();
  this.btnRestoreHuffman2 = new System.Windows.Forms.Button();
  this.btnRestoreHuffman1 = new System.Windows.Forms.Button();
  this.txtHuffmanOriginal4 = new System.Windows.Forms.TextBox();
  this.lblHuffmanOriginalMarker4 = new System.Windows.Forms.Label();
  this.lblHuffmanOriginal4 = new System.Windows.Forms.Label();
  this.txtHuffman4 = new System.Windows.Forms.TextBox();
  this.lblHuffmanMarker4 = new System.Windows.Forms.Label();
  this.lblHuffman4 = new System.Windows.Forms.Label();
  this.txtHuffmanOriginal2 = new System.Windows.Forms.TextBox();
  this.lblHuffmanOriginalMarker2 = new System.Windows.Forms.Label();
  this.lblHuffmanOriginal2 = new System.Windows.Forms.Label();
  this.txtHuffman2 = new System.Windows.Forms.TextBox();
  this.lblHuffmanMarker2 = new System.Windows.Forms.Label();
  this.lblHuffman2 = new System.Windows.Forms.Label();
  this.txtHuffmanOriginal3 = new System.Windows.Forms.TextBox();
  this.lblHuffmanOriginalMarker3 = new System.Windows.Forms.Label();
  this.lblHuffmanOriginal3 = new System.Windows.Forms.Label();
  this.txtHuffman3 = new System.Windows.Forms.TextBox();
```

```
this.lblHuffmanMarker3 = new System.Windows.Forms.Label();
this.lblHuffman3 = new System.Windows.Forms.Label();
this.txtHuffmanOriginal1 = new System.Windows.Forms.TextBox();
this.lblHuffmanOriginalMarker1 = new System.Windows.Forms.Label();
this.lblHuffmanOriginal1 = new System.Windows.Forms.Label();
this.txtHuffman1 = new System.Windows.Forms.TextBox();
this.lblHuffmanMarker1 = new System.Windows.Forms.Label();
this.lblHuffman1 = new System.Windows.Forms.Label();
this.tabHuffman2 = new System.Windows.Forms.TabPage();
this.btnClearHuffman8 = new System.Windows.Forms.Button();
this.btnAddRandomHuffman8 = new System.Windows.Forms.Button();
this.btnClearHuffman7 = new System.Windows.Forms.Button();
this.btnAddRandomHuffman7 = new System.Windows.Forms.Button();
this.btnClearHuffman6 = new System.Windows.Forms.Button();
this.btnAddRandomHuffman6 = new System.Windows.Forms.Button();
this.btnClearHuffman5 = new System.Windows.Forms.Button();
this.btnAddRandomHuffman5 = new System.Windows.Forms.Button();
this.btnRestoreHuffman8 = new System.Windows.Forms.Button();
this.btnRestoreHuffman7 = new System.Windows.Forms.Button();
this.btnRestoreHuffman6 = new System.Windows.Forms.Button();
this.btnRestoreHuffman5 = new System.Windows.Forms.Button();
this.txtHuffmanOriginal8 = new System.Windows.Forms.TextBox();
this.lblHuffmanOriginalMarker8 = new System.Windows.Forms.Label();
this.lblHuffmanOriginal8 = new System.Windows.Forms.Label();
this.txtHuffman8 = new System.Windows.Forms.TextBox();
this.lblHuffmanMarker8 = new System.Windows.Forms.Label();
this.lblHuffman8 = new System.Windows.Forms.Label();
this.txtHuffmanOriginal6 = new System.Windows.Forms.TextBox();
this.lblHuffmanOriginalMarker6 = new System.Windows.Forms.Label();
this.lblHuffmanOriginal6 = new System.Windows.Forms.Label();
this.txtHuffman6 = new System.Windows.Forms.TextBox();
this.lblHuffmanMarker6 = new System.Windows.Forms.Label();
this.lblHuffman6 = new System.Windows.Forms.Label();
this.txtHuffmanOriginal7 = new System.Windows.Forms.TextBox();
this.lblHuffmanOriginalMarker7 = new System.Windows.Forms.Label();
this.lblHuffmanOriginal7 = new System.Windows.Forms.Label();
this.txtHuffman7 = new System.Windows.Forms.TextBox();
this.lblHuffmanMarker7 = new System.Windows.Forms.Label();
this.lblHuffman7 = new System.Windows.Forms.Label();
this.txtHuffmanOriginal5 = new System.Windows.Forms.TextBox();
this.lblHuffmanOriginalMarker5 = new System.Windows.Forms.Label();
this.lblHuffmanOriginal5 = new System.Windows.Forms.Label();
this.txtHuffman5 = new System.Windows.Forms.TextBox();
this.lblHuffmanMarker5 = new System.Windows.Forms.Label();
this.lblHuffman5 = new System.Windows.Forms.Label();
this.tabQuantizer = new System.Windows.Forms.TabPage();
this.txtQuantizerTableNum4 = new System.Windows.Forms.Label();
this.lblQuantizerTableNum4 = new System.Windows.Forms.Label();
this.txtQuantizerTableNum3 = new System.Windows.Forms.Label();
this.lblQuantizerTableNum3 = new System.Windows.Forms.Label();
this.txtQuantizerTableNum2 = new System.Windows.Forms.Label();
this.lblQuantizerTableNum2 = new System.Windows.Forms.Label();
this.txtQuantizerTableNum1 = new System.Windows.Forms.Label();
this.lblQuantizerTableNum1 = new System.Windows.Forms.Label();
this.btnClearQuantizer4 = new System.Windows.Forms.Button();
this.btnAddRandomQuantizer4 = new System.Windows.Forms.Button();
this.btnClearQuantizer3 = new System.Windows.Forms.Button();
this.btnAddRandomQuantizer3 = new System.Windows.Forms.Button();
this.btnClearQuantizer2 = new System.Windows.Forms.Button();
this.btnAddRandomQuantizer2 = new System.Windows.Forms.Button();
this.btnClearQuantizer1 = new System.Windows.Forms.Button();
this.btnAddRandomQuantizer1 = new System.Windows.Forms.Button();
this.btnRestoreQuantizer4 = new System.Windows.Forms.Button();
this.btnRestoreQuantizer3 = new System.Windows.Forms.Button();
this.btnRestoreQuantizer2 = new System.Windows.Forms.Button();
this.btnRestoreQuantizer1 = new System.Windows.Forms.Button();
this.txtQuantizerOriginal4 = new System.Windows.Forms.TextBox();
this.lblQuantizerOriginalMarker4 = new System.Windows.Forms.Label();
this.lblQuantizerOriginal4 = new System.Windows.Forms.Label();
```



```
this.txtQuantizer4 = new System.Windows.Forms.TextBox();
this.lblQuantizerMarker4 = new System.Windows.Forms.Label();
this.lblQuantizer4 = new System.Windows.Forms.Label();
this.txtQuantizerOriginal2 = new System.Windows.Forms.TextBox();
this.lblQuantizerOriginalMarker2 = new System.Windows.Forms.Label();
this.lblQuantizerOriginal2 = new System.Windows.Forms.Label();
this.txtQuantizer2 = new System.Windows.Forms.TextBox();
this.lblQuantizerMarker2 = new System.Windows.Forms.Label();
this.lblQuantizer2 = new System.Windows.Forms.Label();
this.txtQuantizerOriginal3 = new System.Windows.Forms.TextBox();
this.lblQuantizerOriginalMarker3 = new System.Windows.Forms.Label();
this.lblQuantizerOriginal3 = new System.Windows.Forms.Label();
this.txtQuantizer3 = new System.Windows.Forms.TextBox();
this.lblQuantizerMarker3 = new System.Windows.Forms.Label();
this.lblQuantizer3 = new System.Windows.Forms.Label();
this.txtQuantizerOriginal1 = new System.Windows.Forms.TextBox();
this.lblQuantizerOriginalMarker1 = new System.Windows.Forms.Label();
this.lblQuantizerOriginal1 = new System.Windows.Forms.Label();
this.txtQuantizer1 = new System.Windows.Forms.TextBox();
this.lblQuantizerMarker1 = new System.Windows.Forms.Label();
this.lblQuantizer1 = new System.Windows.Forms.Label();
this.tabEncodedData = new System.Windows.Forms.TabPage();
this.lblOriginalHeader = new System.Windows.Forms.Label();
this.txtOriginalHeader = new System.Windows.Forms.TextBox();
this.lblScanHeader = new System.Windows.Forms.Label();
this.txtScanHeader = new System.Windows.Forms.TextBox();
this.txtOriginalEncodedData = new System.Windows.Forms.TextBox();
this.lblOriginalEncodedData = new System.Windows.Forms.Label();
this.txtEncodedData = new System.Windows.Forms.TextBox();
this.lblEncodedData = new System.Windows.Forms.Label();
this.tabApplicationData = new System.Windows.Forms.TabPage();
this.txtApplicationData10 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker10 = new System.Windows.Forms.Label();
this.lblApplicationData10 = new System.Windows.Forms.Label();
this.txtApplicationData9 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker9 = new System.Windows.Forms.Label();
this.lblApplicationData9 = new System.Windows.Forms.Label();
this.txtApplicationData8 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker8 = new System.Windows.Forms.Label();
this.lblApplicationData8 = new System.Windows.Forms.Label();
this.txtApplicationData7 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker7 = new System.Windows.Forms.Label();
this.lblApplicationData7 = new System.Windows.Forms.Label();
this.txtApplicationData6 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker6 = new System.Windows.Forms.Label();
this.lblApplicationData6 = new System.Windows.Forms.Label();
this.txtApplicationData5 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker5 = new System.Windows.Forms.Label();
this.lblApplicationData5 = new System.Windows.Forms.Label();
this.txtApplicationData4 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker4 = new System.Windows.Forms.Label();
this.lblApplicationData4 = new System.Windows.Forms.Label();
this.txtApplicationData3 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker3 = new System.Windows.Forms.Label();
this.lblApplicationData3 = new System.Windows.Forms.Label();
this.txtApplicationData2 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker2 = new System.Windows.Forms.Label();
this.lblApplicationData2 = new System.Windows.Forms.Label();
this.txtApplicationData1 = new System.Windows.Forms.TextBox();
this.lblApplicationMarker1 = new System.Windows.Forms.Label();
this.lblApplicationData1 = new System.Windows.Forms.Label();
this.tabMisc = new System.Windows.Forms.TabPage();
this.lblExpandMarker = new System.Windows.Forms.Label();
this.txtExpand = new System.Windows.Forms.TextBox();
this.lblExpand = new System.Windows.Forms.Label();
this.txtHierarchial = new System.Windows.Forms.TextBox();
this.lblHierarchialMarker = new System.Windows.Forms.Label();
this.lblHierarchial = new System.Windows.Forms.Label();
this.txtRestartMod8 = new System.Windows.Forms.TextBox();
```

```
    this.lblRestartMod8 = new System.Windows.Forms.Label();
    this.txtError = new System.Windows.Forms.TextBox();
    this.lblError = new System.Windows.Forms.Label();
    this.lblNumberLinesMarker = new System.Windows.Forms.Label();
    this.lblRestartMarker = new System.Windows.Forms.Label();
    this.txtNumberLines = new System.Windows.Forms.TextBox();
    this.lblNumberLines = new System.Windows.Forms.Label();
    this.txtRestart = new System.Windows.Forms.TextBox();
    this.lblRestart = new System.Windows.Forms.Label();
    this.picManipulatedSmall = new System.Windows.Forms.PictureBox();
    this.picOriginalSmall = new System.Windows.Forms.PictureBox();
    this.tabOriginal = new System.Windows.Forms.TabPage();
    this.picOriginal = new System.Windows.Forms.PictureBox();
    this.tabManipulated = new System.Windows.Forms.TabPage();
    this.picManipulated = new System.Windows.Forms.PictureBox();
    this.openFileDialog = new System.Windows.Forms.OpenFileDialog();
    this.toolTips = new System.Windows.Forms.ToolTip(this.components);
    this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();
    this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
    this.timerSplash = new System.Windows.Forms.Timer(this.components);
    this.tabMain.SuspendLayout();
    this.tabConsol.SuspendLayout();
    this.tabSubConsole.SuspendLayout();
    this.tabProject.SuspendLayout();
    this.tabFile.SuspendLayout();
    this.tabHeaders.SuspendLayout();
    this.tabHuffman1.SuspendLayout();
    this.tabHuffman2.SuspendLayout();
    this.tabQuantizer.SuspendLayout();
    this.tabEncodedData.SuspendLayout();
    this.tabApplicationData.SuspendLayout();
    this.tabMisc.SuspendLayout();
    this.tabOriginal.SuspendLayout();
    this.tabManipulated.SuspendLayout();
    this.SuspendLayout();
    //
    // menuFrmMain
    //
    this.menuFrmMain.MenuItems.AddRange(new
                System.Windows.Forms.MenuItem[] {
                                        this.menuFile,
                                        this.menuEdit,
                                        this.menuView,
                                        this.menuItem2});
    //
    // menuFile
    //
    this.menuFile.Index = 0;
    this.menuFile.MenuItems.AddRange(new
                System.Windows.Forms.MenuItem[] {
                                       this.menuOpen,
                                       this.menuUpdate,
                                       this.menuItem1,
                                       this.menuNewProject,
                                       this.menuOpenProject,
                                       this.menuSaveProject,
                                       this.menuItem3,
                                       this.menuExit});
    this.menuFile.Text = "&File";
    //
    // menuOpen
    //
    this.menuOpen.Index = 0;
    this.menuOpen.Text = "Loa&d Picture";
    this.menuOpen.Click += new System.EventHandler(this.menuOpen_Click);
    //
    // menuUpdate
    //
    this.menuUpdate.Index = 1;
```



```
    this.menuUpdate.Text = "&Update Picture";
    this.menuUpdate.Click += new
                System.EventHandler(this.menuUpdate_Click);
    //
    // menuItem1
    //
    this.menuItem1.Index = 2;
    this.menuItem1.Text = "-";
    //
    // menuNewProject
    //
    this.menuNewProject.Index = 3;
    this.menuNewProject.Text = "&New Project";
    this.menuNewProject.Click += new
                System.EventHandler(this.menuNewProject_Click);
    //
    // menuOpenProject
    //
    this.menuOpenProject.Index = 4;
    this.menuOpenProject.Text = "Open &Project";
    this.menuOpenProject.Click += new
                System.EventHandler(this.menuOpenProject_Click);
    //
    // menuSaveProject
    //
    this.menuSaveProject.Index = 5;
    this.menuSaveProject.Text = "&Save Project";
    this.menuSaveProject.Click += new
                System.EventHandler(this.menuSaveProject_Click);
    //
    // menuItem3
    //
    this.menuItem3.Index = 6;
    this.menuItem3.Text = "-";
    //
    // menuExit
    //
    this.menuExit.Index = 7;
    this.menuExit.Text = "E&xit";
    this.menuExit.Click += new System.EventHandler(this.menuExit_Click);
    //
    // menuEdit
    //
    this.menuEdit.Index = 1;
    this.menuEdit.MenuItems.AddRange(new
                System.Windows.Forms.MenuItem[] {
                                        this.menuCopy,
                                        this.menuCut,
                                        this.menuPaste});
    this.menuEdit.Text = "&Edit";
    //
    // menuCopy
    //
    this.menuCopy.Index = 0;
    this.menuCopy.Text = "&Copy";
    this.menuCopy.Click += new System.EventHandler(this.menuCopy_Click);
    //
    // menuCut
    //
    this.menuCut.Index = 1;
    this.menuCut.Text = "Cut";
    this.menuCut.Click += new System.EventHandler(this.menuCut_Click);
    //
    // menuPaste
    //
    this.menuPaste.Index = 2;
    this.menuPaste.Text = "Paste";
    this.menuPaste.Click += new System.EventHandler(this.menuPaste_Click);
    //
```

```
        // menuView
        //
        this.menuView.Index = 2;
        this.menuView.MenuItems.AddRange(new
                    System.Windows.Forms.MenuItem[] {
                                            this.menuStretchMode});
        this.menuView.Text = "&View";
        //
        // menuStretchMode
        //
        this.menuStretchMode.Index = 0;
        this.menuStretchMode.MenuItems.AddRange(new
                    System.Windows.Forms.MenuItem[] {
                                              this.menuLargeOriginal,
                                              this.menuLargeManipulated,
                                              this.menuSmallOriginal,
                                              this.menuSmallManipulated,
                                              this.menuAll});
        this.menuStretchMode.Text = "S&tretch Mode";
        //
        // menuLargeOriginal
        //
        this.menuLargeOriginal.Index = 0;
        this.menuLargeOriginal.Text = "Large Original";
        this.menuLargeOriginal.Click += new
                    System.EventHandler(this.menuLargeOriginal_Click);
        //
        // menuLargeManipulated
        //
        this.menuLargeManipulated.Index = 1;
        this.menuLargeManipulated.Text = "Large Manipulated";
        this.menuLargeManipulated.Click += new
                    System.EventHandler(this.menuLargeManipulated_Click);
        //
        // menuSmallOriginal
        //
        this.menuSmallOriginal.Index = 2;
        this.menuSmallOriginal.Text = "Small Original";
        this.menuSmallOriginal.Click += new
                    System.EventHandler(this.menuSmallOriginal_Click);
        //
        // menuSmallManipulated
        //
        this.menuSmallManipulated.Index = 3;
        this.menuSmallManipulated.Text = "Small Manipulated";
        this.menuSmallManipulated.Click += new
                    System.EventHandler(this.menuSmallManipulated_Click);
        //
        // menuAll
        //
        this.menuAll.Index = 4;
        this.menuAll.Text = "A&LL Pictures";
        this.menuAll.Click += new System.EventHandler(this.menuAll_Click);
        //
        // menuItem2
        //
        this.menuItem2.Index = 3;
        this.menuItem2.MenuItems.AddRange(new
                    System.Windows.Forms.MenuItem[] {
                                              this.menuTutorial,
                                              this.menuManual,
                                              this.menuItem6,
                                              this.menuAbout});
        this.menuItem2.Text = "&Help";
        //
        // menuTutorial
        //
        this.menuTutorial.Index = 0;
        this.menuTutorial.Text = "Tutorial";
```



```
        this.menuTutorial.Click += new
                    System.EventHandler(this.menuTutorial_Click);
        //
        // menuManual
        //
        this.menuManual.Index = 1;
        this.menuManual.Text = "Manual";
        this.menuManual.Click += new
                    System.EventHandler(this.menuManual_Click);
        //
        // menuItem6
        //
        this.menuItem6.Index = 2;
        this.menuItem6.Text = "-";
        //
        // menuAbout
        //
        this.menuAbout.Index = 3;
        this.menuAbout.Text = "About";
        this.menuAbout.Click += new System.EventHandler(this.menuAbout_Click);
        //
        // tabMain
        //
        this.tabMain.Controls.Add(this.tabConsol);
        this.tabMain.Controls.Add(this.tabOriginal);
        this.tabMain.Controls.Add(this.tabManipulated);
        this.tabMain.Dock = System.Windows.Forms.DockStyle.Fill;
        this.tabMain.Location = new System.Drawing.Point(0, 0);
        this.tabMain.Name = "tabMain";
        this.tabMain.SelectedIndex = 0;
        this.tabMain.Size = new System.Drawing.Size(904, 653);
        this.tabMain.TabIndex = 0;
        //
        // tabConsol
        //
        this.tabConsol.Controls.Add(this.tabSubConsole);
        this.tabConsol.Controls.Add(this.picManipulatedSmall);
        this.tabConsol.Controls.Add(this.picOriginalSmall);
        this.tabConsol.Location = new System.Drawing.Point(4, 22);
        this.tabConsol.Name = "tabConsol";
        this.tabConsol.Size = new System.Drawing.Size(896, 627);
        this.tabConsol.TabIndex = 0;
        this.tabConsol.Text = "Console";
        //
        // tabSubConsole
        //
        this.tabSubConsole.Controls.Add(this.tabProject);
        this.tabSubConsole.Controls.Add(this.tabFile);
        this.tabSubConsole.Controls.Add(this.tabHeaders);
        this.tabSubConsole.Controls.Add(this.tabHuffman1);
        this.tabSubConsole.Controls.Add(this.tabHuffman2);
        this.tabSubConsole.Controls.Add(this.tabQuantizer);
        this.tabSubConsole.Controls.Add(this.tabEncodedData);
        this.tabSubConsole.Controls.Add(this.tabApplicationData);
        this.tabSubConsole.Controls.Add(this.tabMisc);
        this.tabSubConsole.Dock = System.Windows.Forms.DockStyle.Bottom;
        this.tabSubConsole.ItemSize = new System.Drawing.Size(45, 18);
        this.tabSubConsole.Location = new System.Drawing.Point(0, 355);
        this.tabSubConsole.Name = "tabSubConsole";
        this.tabSubConsole.SelectedIndex = 0;
        this.tabSubConsole.Size = new System.Drawing.Size(896, 272);
        this.tabSubConsole.TabIndex = 2;
        //
        // tabProject
        //
        this.tabProject.Controls.Add(this.lblNotes);
        this.tabProject.Controls.Add(this.btnUpdatePicture);
        this.tabProject.Controls.Add(this.btnSavePicture);
        this.tabProject.Controls.Add(this.btnLoadPicture);
```

```
        this.tabProject.Controls.Add(this.lblFilePath);
        this.tabProject.Controls.Add(this.txtProjectPath);
        this.tabProject.Controls.Add(this.btnLoad);
        this.tabProject.Controls.Add(this.btnSave);
        this.tabProject.Controls.Add(this.btnNew);
        this.tabProject.Controls.Add(this.txtNotes);
        this.tabProject.Location = new System.Drawing.Point(4, 22);
        this.tabProject.Name = "tabProject";
        this.tabProject.Size = new System.Drawing.Size(888, 246);
        this.tabProject.TabIndex = 10;
        this.tabProject.Text = "Project";
        //
        // lblNotes
        //
        this.lblNotes.Location = new System.Drawing.Point(16, 40);
        this.lblNotes.Name = "lblNotes";
        this.lblNotes.Size = new System.Drawing.Size(80, 16);
        this.lblNotes.TabIndex = 9;
        this.lblNotes.Text = "Project Notes:";
        //
        // btnUpdatePicture
        //
        this.btnUpdatePicture.Location = new System.Drawing.Point(776, 208);
        this.btnUpdatePicture.Name = "btnUpdatePicture";
        this.btnUpdatePicture.Size = new System.Drawing.Size(88, 24);
        this.btnUpdatePicture.TabIndex = 8;
        this.btnUpdatePicture.Text = "Update Picture";
        this.btnUpdatePicture.Click += new
                    System.EventHandler(this.btnUpdatePicture_Click);
        //
        // btnSavePicture
        //
        this.btnSavePicture.Location = new System.Drawing.Point(776, 160);
        this.btnSavePicture.Name = "btnSavePicture";
        this.btnSavePicture.Size = new System.Drawing.Size(88, 24);
        this.btnSavePicture.TabIndex = 7;
        this.btnSavePicture.Text = "Save Picture";
        //
        // btnLoadPicture
        //
        this.btnLoadPicture.Location = new System.Drawing.Point(776, 128);
        this.btnLoadPicture.Name = "btnLoadPicture";
        this.btnLoadPicture.Size = new System.Drawing.Size(88, 24);
        this.btnLoadPicture.TabIndex = 6;
        this.btnLoadPicture.Text = "Load Picture";
        this.btnLoadPicture.Click += new
                    System.EventHandler(this.btnLoadPicture_Click);
        //
        // lblFilePath
        //
        this.lblFilePath.Location = new System.Drawing.Point(16, 8);
        this.lblFilePath.Name = "lblFilePath";
        this.lblFilePath.Size = new System.Drawing.Size(96, 16);
        this.lblFilePath.TabIndex = 5;
        this.lblFilePath.Text = "Project File Path:";
        //
        // txtProjectPath
        //
        this.txtProjectPath.Location = new System.Drawing.Point(112, 8);
        this.txtProjectPath.Name = "txtProjectPath";
        this.txtProjectPath.Size = new System.Drawing.Size(640, 20);
        this.txtProjectPath.TabIndex = 4;
        this.txtProjectPath.Text = "";
        this.toolTips.SetToolTip(this.txtProjectPath,
            "Path to the SEP (Selective Encryption Project) name and path.");
        //
        // btnLoad
        //
        this.btnLoad.Location = new System.Drawing.Point(776, 48);
```



```
        this.btnLoad.Name = "btnLoad";
        this.btnLoad.Size = new System.Drawing.Size(88, 24);
        this.btnLoad.TabIndex = 3;
        this.btnLoad.Text = "Open Project";
        this.btnLoad.Click += new System.EventHandler(this.btnLoad_Click);
        //
        // btnSave
        //
        this.btnSave.Location = new System.Drawing.Point(776, 80);
        this.btnSave.Name = "btnSave";
        this.btnSave.Size = new System.Drawing.Size(88, 24);
        this.btnSave.TabIndex = 2;
        this.btnSave.Text = "Save Project";
        this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
        //
        // btnNew
        //
        this.btnNew.Location = new System.Drawing.Point(776, 16);
        this.btnNew.Name = "btnNew";
        this.btnNew.Size = new System.Drawing.Size(88, 24);
        this.btnNew.TabIndex = 1;
        this.btnNew.Text = "New Project";
        this.btnNew.Click += new System.EventHandler(this.btnNew_Click);
        //
        // txtNotes
        //
        this.txtNotes.AcceptsTab = true;
        this.txtNotes.Location = new System.Drawing.Point(112, 40);
        this.txtNotes.Multiline = true;
        this.txtNotes.Name = "txtNotes";
        this.txtNotes.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
        this.txtNotes.Size = new System.Drawing.Size(640, 192);
        this.txtNotes.TabIndex = 0;
        this.txtNotes.Text = "";
        this.toolTips.SetToolTip(this.txtNotes,
              "These are the SEP (Selective Encryption Project) notes.");
        //
        // tabFile
        //
        this.tabFile.Controls.Add(this.txtManipulatedFile);
        this.tabFile.Controls.Add(this.lblComments);
        this.tabFile.Controls.Add(this.txtComments);
        this.tabFile.Controls.Add(this.txtFileSize);
        this.tabFile.Controls.Add(this.lblFileSize);
        this.tabFile.Controls.Add(this.lblManipulatedFile);
        this.tabFile.Controls.Add(this.lblOriginalFile);
        this.tabFile.Controls.Add(this.txtOriginalFile);
        this.tabFile.Location = new System.Drawing.Point(4, 22);
        this.tabFile.Name = "tabFile";
        this.tabFile.Size = new System.Drawing.Size(888, 246);
        this.tabFile.TabIndex = 5;
        this.tabFile.Text = "File Information";
        //
        // txtManipulatedFile
        //
        this.txtManipulatedFile.Location = new System.Drawing.Point(128, 48);
        this.txtManipulatedFile.Name = "txtManipulatedFile";
        this.txtManipulatedFile.Size = new System.Drawing.Size(752, 20);
        this.txtManipulatedFile.TabIndex = 0;
        this.txtManipulatedFile.Text = "";
        this.toolTips.SetToolTip(this.txtManipulatedFile,
             "This is the Manipulated file.");
        this.txtManipulatedFile.TextChanged += new
                    System.EventHandler(this.txtManipulatedFile_TextChanged);
        //
        // lblComments
        //
        this.lblComments.Location = new System.Drawing.Point(8, 113);
        this.lblComments.Name = "lblComments";
```

```
      this.lblComments.Size = new System.Drawing.Size(112, 39);
      this.lblComments.TabIndex = 9;
      this.lblComments.Text = "File Comments:        (Not Saved)";
      //
      // txtComments
      //
      this.txtComments.Location = new System.Drawing.Point(128, 113);
      this.txtComments.Multiline = true;
      this.txtComments.Name = "txtComments";
      this.txtComments.ScrollBars =
                  System.Windows.Forms.ScrollBars.Horizontal;
      this.txtComments.Size = new System.Drawing.Size(752, 71);
      this.txtComments.TabIndex = 8;
      this.txtComments.Text = "";
      this.toolTips.SetToolTip(this.txtComments,
            "These are the comments contain within the original file.");
      //
      // txtFileSize
      //
      this.txtFileSize.Location = new System.Drawing.Point(128, 80);
      this.txtFileSize.Name = "txtFileSize";
      this.txtFileSize.Size = new System.Drawing.Size(128, 20);
      this.txtFileSize.TabIndex = 6;
      this.txtFileSize.TabStop = false;
      this.txtFileSize.Text = "0";
      this.toolTips.SetToolTip(this.txtFileSize,
            "This is the size of the original file.");
      //
      // lblFileSize
      //
      this.lblFileSize.Location = new System.Drawing.Point(8, 80);
      this.lblFileSize.Name = "lblFileSize";
      this.lblFileSize.Size = new System.Drawing.Size(96, 16);
      this.lblFileSize.TabIndex = 7;
      this.lblFileSize.Text = "File Size:";
      //
      // lblManipulatedFile
      //
      this.lblManipulatedFile.Location = new System.Drawing.Point(8, 48);
      this.lblManipulatedFile.Name = "lblManipulatedFile";
      this.lblManipulatedFile.Size = new System.Drawing.Size(128, 16);
      this.lblManipulatedFile.TabIndex = 3;
      this.lblManipulatedFile.Text = "Manipulated File Name:";
      //
      // lblOriginalFile
      //
      this.lblOriginalFile.Location = new System.Drawing.Point(8, 16);
      this.lblOriginalFile.Name = "lblOriginalFile";
      this.lblOriginalFile.Size = new System.Drawing.Size(104, 16);
      this.lblOriginalFile.TabIndex = 1;
      this.lblOriginalFile.Text = "Original File Name:";
      //
      // txtOriginalFile
      //
      this.txtOriginalFile.Enabled = false;
      this.txtOriginalFile.Location = new System.Drawing.Point(128, 16);
      this.txtOriginalFile.Name = "txtOriginalFile";
      this.txtOriginalFile.Size = new System.Drawing.Size(752, 20);
      this.txtOriginalFile.TabIndex = 0;
      this.txtOriginalFile.TabStop = false;
      this.txtOriginalFile.Text = "";
      this.toolTips.SetToolTip(this.txtOriginalFile,
            "This is the original file name.");
      //
      // tabHeaders
      //
      this.tabHeaders.Controls.Add(this.lblComponents);
      this.tabHeaders.Controls.Add(this.lblNumberImageComponents);
      this.tabHeaders.Controls.Add(this.lblNumberHuffmanSamples);
```



```
      this.tabHeaders.Controls.Add(this.lblNumberHuffmanLines);
      this.tabHeaders.Controls.Add(this.lblPrecision);
      this.tabHeaders.Controls.Add(this.lblStartHuffmanSize);
      this.tabHeaders.Controls.Add(this.lblStartHuffman);
      this.tabHeaders.Controls.Add(this.txtComponents);
      this.tabHeaders.Controls.Add(this.txtNumberImageComponents);
      this.tabHeaders.Controls.Add(this.txtNumberHuffmanSamples);
      this.tabHeaders.Controls.Add(this.txtNumberHuffmanLines);
      this.tabHeaders.Controls.Add(this.txtPrecision);
      this.tabHeaders.Controls.Add(this.txtStartHuffmanSize);
      this.tabHeaders.Controls.Add(this.txtStartHuffman);
      this.tabHeaders.Location = new System.Drawing.Point(4, 22);
      this.tabHeaders.Name = "tabHeaders";
      this.tabHeaders.Size = new System.Drawing.Size(888, 246);
      this.tabHeaders.TabIndex = 11;
      this.tabHeaders.Text = "Headers";
      //
      // lblComponents
      //
      this.lblComponents.Location = new System.Drawing.Point(168, 48);
      this.lblComponents.Name = "lblComponents";
      this.lblComponents.Size = new System.Drawing.Size(184, 16);
      this.lblComponents.TabIndex = 27;
      this.lblComponents.Text = "Components:";
      //
      // lblNumberImageComponents
      //
      this.lblNumberImageComponents.Location = new
                  System.Drawing.Point(168, 16);
      this.lblNumberImageComponents.Name = "lblNumberImageComponents";
      this.lblNumberImageComponents.Size = new
                  System.Drawing.Size(120, 16);
      this.lblNumberImageComponents.TabIndex = 26;
      this.lblNumberImageComponents.Text = "Number Components:";
      //
      // lblNumberHuffmanSamples
      //
      this.lblNumberHuffmanSamples.Location = new
                  System.Drawing.Point(8, 176);
      this.lblNumberHuffmanSamples.Name = "lblNumberHuffmanSamples";
      this.lblNumberHuffmanSamples.Size = new System.Drawing.Size(56, 16);
      this.lblNumberHuffmanSamples.TabIndex = 25;
      this.lblNumberHuffmanSamples.Text = "Width:";
      this.toolTips.SetToolTip(this.lblNumberHuffmanSamples,
            "The number of samples per line in the Huffman.");
      //
      // lblNumberHuffmanLines
      //
      this.lblNumberHuffmanLines.Location = new System.Drawing.Point(8, 136);
      this.lblNumberHuffmanLines.Name = "lblNumberHuffmanLines";
      this.lblNumberHuffmanLines.Size = new System.Drawing.Size(56, 16);
      this.lblNumberHuffmanLines.TabIndex = 24;
      this.lblNumberHuffmanLines.Text = "Height:";
      this.toolTips.SetToolTip(this.lblNumberHuffmanLines,
            "Number of lines in the source");
      //
      // lblPrecision
      //
      this.lblPrecision.Location = new System.Drawing.Point(8, 96);
      this.lblPrecision.Name = "lblPrecision";
      this.lblPrecision.Size = new System.Drawing.Size(56, 16);
      this.lblPrecision.TabIndex = 23;
      this.lblPrecision.Text = "Precision:";
      this.toolTips.SetToolTip(this.lblPrecision,
            "Precision in the Huffman");
      //
      // lblStartHuffmanSize
      //
      this.lblStartHuffmanSize.Location = new System.Drawing.Point(8, 56);
```

```
      this.lblStartHuffmanSize.Name = "lblStartHuffmanSize";
      this.lblStartHuffmanSize.Size = new System.Drawing.Size(56, 16);
      this.lblStartHuffmanSize.TabIndex = 22;
      this.lblStartHuffmanSize.Text = "Size:";
      this.toolTips.SetToolTip(this.lblStartHuffmanSize,
            "Size of the Huffman header size.");
      //
      // lblStartHuffman
      //
      this.lblStartHuffman.Location = new System.Drawing.Point(8, 16);
      this.lblStartHuffman.Name = "lblStartHuffman";
      this.lblStartHuffman.Size = new System.Drawing.Size(56, 16);
      this.lblStartHuffman.TabIndex = 21;
      this.lblStartHuffman.Text = "Marker:";
      this.toolTips.SetToolTip(this.lblStartHuffman,
            "Value of the Huffman marker.");
      //
      // txtComponents
      //
      this.txtComponents.AcceptsTab = true;
      this.txtComponents.Location = new System.Drawing.Point(168, 64);
      this.txtComponents.MaxLength = 1024;
      this.txtComponents.Name = "txtComponents";
      this.txtComponents.ScrollBars =
                  System.Windows.Forms.RichTextBoxScrollBars.Vertical;
      this.txtComponents.Size = new System.Drawing.Size(208, 152);
      this.txtComponents.TabIndex = 20;
      this.txtComponents.Text = "";
      //
      // txtNumberImageComponents
      //
      this.txtNumberImageComponents.Location = new
                  System.Drawing.Point(296, 16);
      this.txtNumberImageComponents.MaxLength = 32;
      this.txtNumberImageComponents.Name = "txtNumberImageComponents";
      this.txtNumberImageComponents.Size = new System.Drawing.Size(56, 20);
      this.txtNumberImageComponents.TabIndex = 19;
      this.txtNumberImageComponents.Text = "";
      //
      // txtNumberHuffmanSamples
      //
      this.txtNumberHuffmanSamples.Location = new
                  System.Drawing.Point(80, 176);
      this.txtNumberHuffmanSamples.MaxLength = 32;
      this.txtNumberHuffmanSamples.Name = "txtNumberHuffmanSamples";
      this.txtNumberHuffmanSamples.Size = new System.Drawing.Size(56, 20);
      this.txtNumberHuffmanSamples.TabIndex = 18;
      this.txtNumberHuffmanSamples.Text = "";
      //
      // txtNumberHuffmanLines
      //
      this.txtNumberHuffmanLines.Location = new
                  System.Drawing.Point(80, 136);
      this.txtNumberHuffmanLines.MaxLength = 32;
      this.txtNumberHuffmanLines.Name = "txtNumberHuffmanLines";
      this.txtNumberHuffmanLines.Size = new System.Drawing.Size(56, 20);
      this.txtNumberHuffmanLines.TabIndex = 17;
      this.txtNumberHuffmanLines.Text = "";
      //
      // txtPrecision
      //
      this.txtPrecision.Location = new System.Drawing.Point(80, 96);
      this.txtPrecision.MaxLength = 2048;
      this.txtPrecision.Name = "txtPrecision";
      this.txtPrecision.Size = new System.Drawing.Size(56, 20);
      this.txtPrecision.TabIndex = 16;
      this.txtPrecision.Text = "";
      //
      // txtStartHuffmanSize
```



```
      //
      this.txtStartHuffmanSize.Location = new System.Drawing.Point(80, 56);
      this.txtStartHuffmanSize.MaxLength = 32;
      this.txtStartHuffmanSize.Name = "txtStartHuffmanSize";
      this.txtStartHuffmanSize.Size = new System.Drawing.Size(56, 20);
      this.txtStartHuffmanSize.TabIndex = 15;
      this.txtStartHuffmanSize.Text = "";
      //
      // txtStartHuffman
      //
      this.txtStartHuffman.Location = new System.Drawing.Point(80, 16);
      this.txtStartHuffman.MaxLength = 32;
      this.txtStartHuffman.Name = "txtStartHuffman";
      this.txtStartHuffman.Size = new System.Drawing.Size(56, 20);
      this.txtStartHuffman.TabIndex = 14;
      this.txtStartHuffman.Text = "";
      //
      // tabHuffman1
      //
      this.tabHuffman1.Controls.Add(this.btnClearHuffman4);
      this.tabHuffman1.Controls.Add(this.btnAddRandomHuffman4);
      this.tabHuffman1.Controls.Add(this.btnClearHuffman2);
      this.tabHuffman1.Controls.Add(this.btnAddRandomHuffman2);
      this.tabHuffman1.Controls.Add(this.btnClearHuffman3);
      this.tabHuffman1.Controls.Add(this.btnAddRandomHuffman3);
      this.tabHuffman1.Controls.Add(this.btnClearHuffman1);
      this.tabHuffman1.Controls.Add(this.btnAddRandomHuffman1);
      this.tabHuffman1.Controls.Add(this.btnRestoreHuffman4);
      this.tabHuffman1.Controls.Add(this.btnRestoreHuffman3);
      this.tabHuffman1.Controls.Add(this.btnRestoreHuffman2);
      this.tabHuffman1.Controls.Add(this.btnRestoreHuffman1);
      this.tabHuffman1.Controls.Add(this.txtHuffmanOriginal4);
      this.tabHuffman1.Controls.Add(this.lblHuffmanOriginalMarker4);
      this.tabHuffman1.Controls.Add(this.lblHuffmanOriginal4);
      this.tabHuffman1.Controls.Add(this.txtHuffman4);
      this.tabHuffman1.Controls.Add(this.lblHuffmanMarker4);
      this.tabHuffman1.Controls.Add(this.lblHuffman4);
      this.tabHuffman1.Controls.Add(this.txtHuffmanOriginal2);
      this.tabHuffman1.Controls.Add(this.lblHuffmanOriginalMarker2);
      this.tabHuffman1.Controls.Add(this.lblHuffmanOriginal2);
      this.tabHuffman1.Controls.Add(this.txtHuffman2);
      this.tabHuffman1.Controls.Add(this.lblHuffmanMarker2);
      this.tabHuffman1.Controls.Add(this.lblHuffman2);
      this.tabHuffman1.Controls.Add(this.txtHuffmanOriginal3);
      this.tabHuffman1.Controls.Add(this.lblHuffmanOriginalMarker3);
      this.tabHuffman1.Controls.Add(this.lblHuffmanOriginal3);
      this.tabHuffman1.Controls.Add(this.txtHuffman3);
      this.tabHuffman1.Controls.Add(this.lblHuffmanMarker3);
      this.tabHuffman1.Controls.Add(this.lblHuffman3);
      this.tabHuffman1.Controls.Add(this.txtHuffmanOriginal1);
      this.tabHuffman1.Controls.Add(this.lblHuffmanOriginalMarker1);
      this.tabHuffman1.Controls.Add(this.lblHuffmanOriginal1);
      this.tabHuffman1.Controls.Add(this.txtHuffman1);
      this.tabHuffman1.Controls.Add(this.lblHuffmanMarker1);
      this.tabHuffman1.Controls.Add(this.lblHuffman1);
      this.tabHuffman1.Location = new System.Drawing.Point(4, 22);
      this.tabHuffman1.Name = "tabHuffman1";
      this.tabHuffman1.Size = new System.Drawing.Size(888, 246);
      this.tabHuffman1.TabIndex = 0;
      this.tabHuffman1.Text = "Huffman Tables 1";
      //
      // btnClearHuffman4
      //
      this.btnClearHuffman4.Font = new
                  System.Drawing.Font(
            "Microsoft Sans Serif", 7F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.btnClearHuffman4.Location = new System.Drawing.Point(448, 152);
      this.btnClearHuffman4.Name = "btnClearHuffman4";
```

```
this.btnClearHuffman4.Size = new System.Drawing.Size(40, 16);
this.btnClearHuffman4.TabIndex = 63;
this.btnClearHuffman4.Text = "Clear";
this.btnClearHuffman4.Click += new
            System.EventHandler(this.btnClearHuffman4_Click);
//
// btnAddRandomHuffman4
//
this.btnAddRandomHuffman4.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
               System.Drawing.FontStyle.Regular,
               System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnAddRandomHuffman4.Location = new
            System.Drawing.Point(496, 152);
this.btnAddRandomHuffman4.Name = "btnAddRandomHuffman4";
this.btnAddRandomHuffman4.Size = new System.Drawing.Size(48, 16);
this.btnAddRandomHuffman4.TabIndex = 62;
this.btnAddRandomHuffman4.Text = "Random";
this.btnAddRandomHuffman4.Click += new
            System.EventHandler(this.btnAddRandomHuffman4_Click);
//
// btnClearHuffman2
//
this.btnClearHuffman2.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnClearHuffman2.Location = new System.Drawing.Point(448, 32);
this.btnClearHuffman2.Name = "btnClearHuffman2";
this.btnClearHuffman2.Size = new System.Drawing.Size(40, 16);
this.btnClearHuffman2.TabIndex = 61;
this.btnClearHuffman2.Text = "Clear";
this.btnClearHuffman2.Click += new
            System.EventHandler(this.btnClearHuffman2_Click);
//
// btnAddRandomHuffman2
//
this.btnAddRandomHuffman2.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnAddRandomHuffman2.Location = new System.Drawing.Point(496, 32);
this.btnAddRandomHuffman2.Name = "btnAddRandomHuffman2";
this.btnAddRandomHuffman2.Size = new System.Drawing.Size(48, 16);
this.btnAddRandomHuffman2.TabIndex = 60;
this.btnAddRandomHuffman2.Text = "Random";
this.btnAddRandomHuffman2.Click += new
            System.EventHandler(this.btnAddRandomHuffman2_Click);
//
// btnClearHuffman3
//
this.btnClearHuffman3.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnClearHuffman3.Location = new System.Drawing.Point(8, 152);
this.btnClearHuffman3.Name = "btnClearHuffman3";
this.btnClearHuffman3.Size = new System.Drawing.Size(40, 16);
this.btnClearHuffman3.TabIndex = 59;
this.btnClearHuffman3.Text = "Clear";
this.btnClearHuffman3.Click += new
            System.EventHandler(this.btnClearHuffman3_Click);
//
// btnAddRandomHuffman3
//
this.btnAddRandomHuffman3.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
```



```
this.btnAddRandomHuffman3.Location = new System.Drawing.Point(56, 152);
this.btnAddRandomHuffman3.Name = "btnAddRandomHuffman3";
this.btnAddRandomHuffman3.Size = new System.Drawing.Size(48, 16);
this.btnAddRandomHuffman3.TabIndex = 58;
this.btnAddRandomHuffman3.Text = "Random";
this.btnAddRandomHuffman3.Click += new
            System.EventHandler(this.btnAddRandomHuffman3_Click);
//
// btnClearHuffman1
//
this.btnClearHuffman1.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnClearHuffman1.Location = new System.Drawing.Point(8, 32);
this.btnClearHuffman1.Name = "btnClearHuffman1";
this.btnClearHuffman1.Size = new System.Drawing.Size(40, 16);
this.btnClearHuffman1.TabIndex = 57;
this.btnClearHuffman1.Text = "Clear";
this.btnClearHuffman1.Click += new
            System.EventHandler(this.btnClearHuffman1_Click);
//
// btnAddRandomHuffman1
//
this.btnAddRandomHuffman1.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnAddRandomHuffman1.Location = new System.Drawing.Point(56, 32);
this.btnAddRandomHuffman1.Name = "btnAddRandomHuffman1";
this.btnAddRandomHuffman1.Size = new System.Drawing.Size(48, 16);
this.btnAddRandomHuffman1.TabIndex = 56;
this.btnAddRandomHuffman1.Text = "Random";
this.btnAddRandomHuffman1.Click += new
            System.EventHandler(this.btnAddRandomHuffman1_Click);
//
// btnRestoreHuffman4
//
this.btnRestoreHuffman4.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnRestoreHuffman4.Location = new System.Drawing.Point(496, 208);
this.btnRestoreHuffman4.Name = "btnRestoreHuffman4";
this.btnRestoreHuffman4.Size = new System.Drawing.Size(48, 16);
this.btnRestoreHuffman4.TabIndex = 55;
this.btnRestoreHuffman4.Text = "Restore";
this.btnRestoreHuffman4.Click += new
            System.EventHandler(this.btnRestoreHuffman4_Click);
//
// btnRestoreHuffman3
//
this.btnRestoreHuffman3.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnRestoreHuffman3.Location = new System.Drawing.Point(56, 208);
this.btnRestoreHuffman3.Name = "btnRestoreHuffman3";
this.btnRestoreHuffman3.Size = new System.Drawing.Size(48, 16);
this.btnRestoreHuffman3.TabIndex = 54;
this.btnRestoreHuffman3.Text = "Restore";
this.btnRestoreHuffman3.Click += new
            System.EventHandler(this.btnRestoreHuffman3_Click);
//
// btnRestoreHuffman2
//
this.btnRestoreHuffman2.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
```

```
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.btnRestoreHuffman2.Location = new System.Drawing.Point(496, 88);
            this.btnRestoreHuffman2.Name = "btnRestoreHuffman2";
            this.btnRestoreHuffman2.Size = new System.Drawing.Size(48, 16);
            this.btnRestoreHuffman2.TabIndex = 53;
            this.btnRestoreHuffman2.Text = "Restore";
            this.btnRestoreHuffman2.Click += new
                        System.EventHandler(this.btnRestoreHuffman2_Click);
            //
            // btnRestoreHuffman1
            //
            this.btnRestoreHuffman1.Font = new
                        System.Drawing.Font("Microsoft Sans Serif", 7F,
                            System.Drawing.FontStyle.Regular,
                            System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.btnRestoreHuffman1.Location = new System.Drawing.Point(56, 88);
            this.btnRestoreHuffman1.Name = "btnRestoreHuffman1";
            this.btnRestoreHuffman1.Size = new System.Drawing.Size(48, 16);
            this.btnRestoreHuffman1.TabIndex = 52;
            this.btnRestoreHuffman1.Text = "Restore";
            this.btnRestoreHuffman1.Click += new
                        System.EventHandler(this.btnRestoreHuffman1_Click);
            //
            // txtHuffmanOriginal4
            //
            this.txtHuffmanOriginal4.AutoSize = false;
            this.txtHuffmanOriginal4.Enabled = false;
            this.txtHuffmanOriginal4.Location = new System.Drawing.Point(552, 184);
            this.txtHuffmanOriginal4.Multiline = true;
            this.txtHuffmanOriginal4.Name = "txtHuffmanOriginal4";
            this.txtHuffmanOriginal4.ScrollBars =
                        System.Windows.Forms.ScrollBars.Horizontal;
            this.txtHuffmanOriginal4.Size = new System.Drawing.Size(328, 48);
            this.txtHuffmanOriginal4.TabIndex = 26;
            this.txtHuffmanOriginal4.TabStop = false;
            this.txtHuffmanOriginal4.Text = "";
            //
            // lblHuffmanOriginalMarker4
            //
            this.lblHuffmanOriginalMarker4.BackColor =
                        System.Drawing.SystemColors.Window;
            this.lblHuffmanOriginalMarker4.Enabled = false;
            this.lblHuffmanOriginalMarker4.Location = new
                        System.Drawing.Point(512, 184);
            this.lblHuffmanOriginalMarker4.Name = "lblHuffmanOriginalMarker4";
            this.lblHuffmanOriginalMarker4.Size = new System.Drawing.Size(32, 16);
            this.lblHuffmanOriginalMarker4.TabIndex = 25;
            this.lblHuffmanOriginalMarker4.TextAlign =
                        System.Drawing.ContentAlignment.TopCenter;
            //
            // lblHuffmanOriginal4
            //
            this.lblHuffmanOriginal4.Location = new System.Drawing.Point(456, 184);
            this.lblHuffmanOriginal4.Name = "lblHuffmanOriginal4";
            this.lblHuffmanOriginal4.Size = new System.Drawing.Size(64, 16);
            this.lblHuffmanOriginal4.TabIndex = 24;
            this.lblHuffmanOriginal4.Text = "Original 4:";
            //
            // txtHuffman4
            //
            this.txtHuffman4.AutoSize = false;
            this.txtHuffman4.Location = new System.Drawing.Point(552, 128);
            this.txtHuffman4.Multiline = true;
            this.txtHuffman4.Name = "txtHuffman4";
            this.txtHuffman4.ScrollBars =
                        System.Windows.Forms.ScrollBars.Horizontal;
            this.txtHuffman4.Size = new System.Drawing.Size(328, 48);
            this.txtHuffman4.TabIndex = 4;
            this.txtHuffman4.Text = "";
```



```
            this.txtHuffman4.GotFocus += new
                        System.EventHandler(this.txtHuffman4_GotFocus);
            //
            // lblHuffmanMarker4
            //
            this.lblHuffmanMarker4.BackColor = System.Drawing.SystemColors.Window;
            this.lblHuffmanMarker4.Enabled = false;
            this.lblHuffmanMarker4.Location = new System.Drawing.Point(512, 128);
            this.lblHuffmanMarker4.Name = "lblHuffmanMarker4";
            this.lblHuffmanMarker4.Size = new System.Drawing.Size(32, 16);
            this.lblHuffmanMarker4.TabIndex = 22;
            this.lblHuffmanMarker4.TextAlign =
                        System.Drawing.ContentAlignment.TopCenter;
            //
            // lblHuffman4
            //
            this.lblHuffman4.Location = new System.Drawing.Point(456, 128);
            this.lblHuffman4.Name = "lblHuffman4";
            this.lblHuffman4.Size = new System.Drawing.Size(64, 16);
            this.lblHuffman4.TabIndex = 21;
            this.lblHuffman4.Text = "Huffman 4:";
            //
            // txtHuffmanOriginal2
            //
            this.txtHuffmanOriginal2.AutoSize = false;
            this.txtHuffmanOriginal2.Enabled = false;
            this.txtHuffmanOriginal2.Location = new System.Drawing.Point(552, 64);
            this.txtHuffmanOriginal2.Multiline = true;
            this.txtHuffmanOriginal2.Name = "txtHuffmanOriginal2";
            this.txtHuffmanOriginal2.ScrollBars =
                        System.Windows.Forms.ScrollBars.Horizontal;
            this.txtHuffmanOriginal2.Size = new System.Drawing.Size(328, 48);
            this.txtHuffmanOriginal2.TabIndex = 20;
            this.txtHuffmanOriginal2.TabStop = false;
            this.txtHuffmanOriginal2.Text = "";
            //
            // lblHuffmanOriginalMarker2
            //
            this.lblHuffmanOriginalMarker2.BackColor =
                        System.Drawing.SystemColors.Window;
            this.lblHuffmanOriginalMarker2.Enabled = false;
            this.lblHuffmanOriginalMarker2.Location = new
                        System.Drawing.Point(512, 64);
            this.lblHuffmanOriginalMarker2.Name = "lblHuffmanOriginalMarker2";
            this.lblHuffmanOriginalMarker2.Size = new System.Drawing.Size(32, 16);
            this.lblHuffmanOriginalMarker2.TabIndex = 19;
            this.lblHuffmanOriginalMarker2.TextAlign =
                        System.Drawing.ContentAlignment.TopCenter;
            //
            // lblHuffmanOriginal2
            //
            this.lblHuffmanOriginal2.Location = new System.Drawing.Point(456, 64);
            this.lblHuffmanOriginal2.Name = "lblHuffmanOriginal2";
            this.lblHuffmanOriginal2.Size = new System.Drawing.Size(64, 16);
            this.lblHuffmanOriginal2.TabIndex = 18;
            this.lblHuffmanOriginal2.Text = "Original 2:";
            //
            // txtHuffman2
            //
            this.txtHuffman2.AutoSize = false;
            this.txtHuffman2.Location = new System.Drawing.Point(552, 8);
            this.txtHuffman2.Multiline = true;
            this.txtHuffman2.Name = "txtHuffman2";
            this.txtHuffman2.ScrollBars =
                        System.Windows.Forms.ScrollBars.Horizontal;
            this.txtHuffman2.Size = new System.Drawing.Size(328, 48);
            this.txtHuffman2.TabIndex = 1;
            this.txtHuffman2.Text = "";
            this.txtHuffman2.GotFocus += new
```

```
            System.EventHandler(this.txtHuffman2_GotFocus);
        //
        // lblHuffmanMarker2
        //
        this.lblHuffmanMarker2.BackColor = System.Drawing.SystemColors.Window;
        this.lblHuffmanMarker2.Enabled = false;
        this.lblHuffmanMarker2.Location = new System.Drawing.Point(512, 8);
        this.lblHuffmanMarker2.Name = "lblHuffmanMarker2";
        this.lblHuffmanMarker2.Size = new System.Drawing.Size(32, 16);
        this.lblHuffmanMarker2.TabIndex = 16;
        this.lblHuffmanMarker2.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
        //
        // lblHuffman2
        //
        this.lblHuffman2.Location = new System.Drawing.Point(456, 8);
        this.lblHuffman2.Name = "lblHuffman2";
        this.lblHuffman2.Size = new System.Drawing.Size(64, 16);
        this.lblHuffman2.TabIndex = 15;
        this.lblHuffman2.Text = "Huffman 2:";
        //
        // txtHuffmanOriginal3
        //
        this.txtHuffmanOriginal3.AutoSize = false;
        this.txtHuffmanOriginal3.Enabled = false;
        this.txtHuffmanOriginal3.Location = new System.Drawing.Point(112, 184);
        this.txtHuffmanOriginal3.Multiline = true;
        this.txtHuffmanOriginal3.Name = "txtHuffmanOriginal3";
        this.txtHuffmanOriginal3.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
        this.txtHuffmanOriginal3.Size = new System.Drawing.Size(328, 48);
        this.txtHuffmanOriginal3.TabIndex = 14;
        this.txtHuffmanOriginal3.TabStop = false;
        this.txtHuffmanOriginal3.Text = "";
        //
        // lblHuffmanOriginalMarker3
        //
        this.lblHuffmanOriginalMarker3.BackColor =
            System.Drawing.SystemColors.Window;
        this.lblHuffmanOriginalMarker3.Enabled = false;
        this.lblHuffmanOriginalMarker3.Location = new
            System.Drawing.Point(72, 184);
        this.lblHuffmanOriginalMarker3.Name = "lblHuffmanOriginalMarker3";
        this.lblHuffmanOriginalMarker3.Size = new System.Drawing.Size(32, 16);
        this.lblHuffmanOriginalMarker3.TabIndex = 13;
        this.lblHuffmanOriginalMarker3.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
        //
        // lblHuffmanOriginal3
        //
        this.lblHuffmanOriginal3.Location = new System.Drawing.Point(16, 184);
        this.lblHuffmanOriginal3.Name = "lblHuffmanOriginal3";
        this.lblHuffmanOriginal3.Size = new System.Drawing.Size(64, 16);
        this.lblHuffmanOriginal3.TabIndex = 12;
        this.lblHuffmanOriginal3.Text = "Original 3:";
        //
        // txtHuffman3
        //
        this.txtHuffman3.AutoSize = false;
        this.txtHuffman3.Location = new System.Drawing.Point(112, 128);
        this.txtHuffman3.Multiline = true;
        this.txtHuffman3.Name = "txtHuffman3";
        this.txtHuffman3.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
        this.txtHuffman3.Size = new System.Drawing.Size(328, 48);
        this.txtHuffman3.TabIndex = 3;
        this.txtHuffman3.Text = "";
        this.txtHuffman3.GotFocus += new
            System.EventHandler(this.txtHuffman3_GotFocus);
```



```
        //
        // lblHuffmanMarker3
        //
        this.lblHuffmanMarker3.BackColor = System.Drawing.SystemColors.Window;
        this.lblHuffmanMarker3.Enabled = false;
        this.lblHuffmanMarker3.Location = new System.Drawing.Point(72, 128);
        this.lblHuffmanMarker3.Name = "lblHuffmanMarker3";
        this.lblHuffmanMarker3.Size = new System.Drawing.Size(32, 16);
        this.lblHuffmanMarker3.TabIndex = 10;
        this.lblHuffmanMarker3.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
        //
        // lblHuffman3
        //
        this.lblHuffman3.Location = new System.Drawing.Point(16, 128);
        this.lblHuffman3.Name = "lblHuffman3";
        this.lblHuffman3.Size = new System.Drawing.Size(64, 16);
        this.lblHuffman3.TabIndex = 9;
        this.lblHuffman3.Text = "Huffman 3:";
        //
        // txtHuffmanOriginal1
        //
        this.txtHuffmanOriginal1.AutoSize = false;
        this.txtHuffmanOriginal1.Enabled = false;
        this.txtHuffmanOriginal1.Location = new System.Drawing.Point(112, 64);
        this.txtHuffmanOriginal1.Multiline = true;
        this.txtHuffmanOriginal1.Name = "txtHuffmanOriginal1";
        this.txtHuffmanOriginal1.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
        this.txtHuffmanOriginal1.Size = new System.Drawing.Size(328, 48);
        this.txtHuffmanOriginal1.TabIndex = 8;
        this.txtHuffmanOriginal1.TabStop = false;
        this.txtHuffmanOriginal1.Text = "";
        //
        // lblHuffmanOriginalMarker1
        //
        this.lblHuffmanOriginalMarker1.BackColor =
            System.Drawing.SystemColors.Window;
        this.lblHuffmanOriginalMarker1.Enabled = false;
        this.lblHuffmanOriginalMarker1.Location = new
            System.Drawing.Point(72, 64);
        this.lblHuffmanOriginalMarker1.Name = "lblHuffmanOriginalMarker1";
        this.lblHuffmanOriginalMarker1.Size = new System.Drawing.Size(32, 16);
        this.lblHuffmanOriginalMarker1.TabIndex = 7;
        this.lblHuffmanOriginalMarker1.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
        //
        // lblHuffmanOriginal1
        //
        this.lblHuffmanOriginal1.Location = new System.Drawing.Point(16, 64);
        this.lblHuffmanOriginal1.Name = "lblHuffmanOriginal1";
        this.lblHuffmanOriginal1.Size = new System.Drawing.Size(64, 16);
        this.lblHuffmanOriginal1.TabIndex = 6;
        this.lblHuffmanOriginal1.Text = "Original 1:";
        //
        // txtHuffman1
        //
        this.txtHuffman1.AutoSize = false;
        this.txtHuffman1.Location = new System.Drawing.Point(112, 8);
        this.txtHuffman1.Multiline = true;
        this.txtHuffman1.Name = "txtHuffman1";
        this.txtHuffman1.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
        this.txtHuffman1.Size = new System.Drawing.Size(328, 48);
        this.txtHuffman1.TabIndex = 0;
        this.txtHuffman1.Text = "";
        this.txtHuffman1.GotFocus += new
            System.EventHandler(this.txtHuffman1_GotFocus);
        //
```

```
    // lblHuffmanMarker1
    //
    this.lblHuffmanMarker1.BackColor = System.Drawing.SystemColors.Window;
    this.lblHuffmanMarker1.Enabled = false;
    this.lblHuffmanMarker1.Location = new System.Drawing.Point(72, 8);
    this.lblHuffmanMarker1.Name = "lblHuffmanMarker1";
    this.lblHuffmanMarker1.Size = new System.Drawing.Size(32, 16);
    this.lblHuffmanMarker1.TabIndex = 1;
    this.lblHuffmanMarker1.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblHuffman1
    //
    this.lblHuffman1.Location = new System.Drawing.Point(16, 8);
    this.lblHuffman1.Name = "lblHuffman1";
    this.lblHuffman1.Size = new System.Drawing.Size(64, 16);
    this.lblHuffman1.TabIndex = 0;
    this.lblHuffman1.Text = "Huffman 1:";
    //
    // tabHuffman2
    //
    this.tabHuffman2.Controls.Add(this.btnClearHuffman8);
    this.tabHuffman2.Controls.Add(this.btnAddRandomHuffman8);
    this.tabHuffman2.Controls.Add(this.btnClearHuffman7);
    this.tabHuffman2.Controls.Add(this.btnAddRandomHuffman7);
    this.tabHuffman2.Controls.Add(this.btnClearHuffman6);
    this.tabHuffman2.Controls.Add(this.btnAddRandomHuffman6);
    this.tabHuffman2.Controls.Add(this.btnClearHuffman5);
    this.tabHuffman2.Controls.Add(this.btnAddRandomHuffman5);
    this.tabHuffman2.Controls.Add(this.btnRestoreHuffman8);
    this.tabHuffman2.Controls.Add(this.btnRestoreHuffman7);
    this.tabHuffman2.Controls.Add(this.btnRestoreHuffman6);
    this.tabHuffman2.Controls.Add(this.btnRestoreHuffman5);
    this.tabHuffman2.Controls.Add(this.txtHuffmanOriginal8);
    this.tabHuffman2.Controls.Add(this.lblHuffmanOriginalMarker8);
    this.tabHuffman2.Controls.Add(this.lblHuffmanOriginal8);
    this.tabHuffman2.Controls.Add(this.txtHuffman8);
    this.tabHuffman2.Controls.Add(this.lblHuffmanMarker8);
    this.tabHuffman2.Controls.Add(this.lblHuffman8);
    this.tabHuffman2.Controls.Add(this.txtHuffmanOriginal6);
    this.tabHuffman2.Controls.Add(this.lblHuffmanOriginalMarker6);
    this.tabHuffman2.Controls.Add(this.lblHuffmanOriginal6);
    this.tabHuffman2.Controls.Add(this.txtHuffman6);
    this.tabHuffman2.Controls.Add(this.lblHuffmanMarker6);
    this.tabHuffman2.Controls.Add(this.lblHuffman6);
    this.tabHuffman2.Controls.Add(this.txtHuffmanOriginal7);
    this.tabHuffman2.Controls.Add(this.lblHuffmanOriginalMarker7);
    this.tabHuffman2.Controls.Add(this.lblHuffmanOriginal7);
    this.tabHuffman2.Controls.Add(this.txtHuffman7);
    this.tabHuffman2.Controls.Add(this.lblHuffmanMarker7);
    this.tabHuffman2.Controls.Add(this.lblHuffman7);
    this.tabHuffman2.Controls.Add(this.txtHuffmanOriginal5);
    this.tabHuffman2.Controls.Add(this.lblHuffmanOriginalMarker5);
    this.tabHuffman2.Controls.Add(this.lblHuffmanOriginal5);
    this.tabHuffman2.Controls.Add(this.txtHuffman5);
    this.tabHuffman2.Controls.Add(this.lblHuffmanMarker5);
    this.tabHuffman2.Controls.Add(this.lblHuffman5);
    this.tabHuffman2.Location = new System.Drawing.Point(4, 22);
    this.tabHuffman2.Name = "tabHuffman2";
    this.tabHuffman2.Size = new System.Drawing.Size(888, 246);
    this.tabHuffman2.TabIndex = 7;
    this.tabHuffman2.Text = "Huffman Tables 2";
    //
    // btnClearHuffman8
    //
    this.btnClearHuffman8.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
```



```
    this.btnClearHuffman8.Location = new System.Drawing.Point(448, 152);
    this.btnClearHuffman8.Name = "btnClearHuffman8";
    this.btnClearHuffman8.Size = new System.Drawing.Size(40, 16);
    this.btnClearHuffman8.TabIndex = 65;
    this.btnClearHuffman8.Text = "Clear";
    this.btnClearHuffman8.Click += new
                System.EventHandler(this.btnClearHuffman8_Click);
    //
    // btnAddRandomHuffman8
    //
    this.btnAddRandomHuffman8.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnAddRandomHuffman8.Location = new
                System.Drawing.Point(496, 152);
    this.btnAddRandomHuffman8.Name = "btnAddRandomHuffman8";
    this.btnAddRandomHuffman8.Size = new System.Drawing.Size(48, 16);
    this.btnAddRandomHuffman8.TabIndex = 64;
    this.btnAddRandomHuffman8.Text = "Random";
    this.btnAddRandomHuffman8.Click += new
                System.EventHandler(this.btnAddRandomHuffman8_Click);
    //
    // btnClearHuffman7
    //
    this.btnClearHuffman7.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnClearHuffman7.Location = new System.Drawing.Point(8, 152);
    this.btnClearHuffman7.Name = "btnClearHuffman7";
    this.btnClearHuffman7.Size = new System.Drawing.Size(40, 16);
    this.btnClearHuffman7.TabIndex = 63;
    this.btnClearHuffman7.Text = "Clear";
    this.btnClearHuffman7.Click += new
                System.EventHandler(this.btnClearHuffman7_Click);
    //
    // btnAddRandomHuffman7
    //
    this.btnAddRandomHuffman7.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnAddRandomHuffman7.Location = new System.Drawing.Point(56, 152);
    this.btnAddRandomHuffman7.Name = "btnAddRandomHuffman7";
    this.btnAddRandomHuffman7.Size = new System.Drawing.Size(48, 16);
    this.btnAddRandomHuffman7.TabIndex = 62;
    this.btnAddRandomHuffman7.Text = "Random";
    this.btnAddRandomHuffman7.Click += new
                System.EventHandler(this.btnAddRandomHuffman7_Click);
    //
    // btnClearHuffman6
    //
    this.btnClearHuffman6.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                     System.Drawing.FontStyle.Regular,
                     System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnClearHuffman6.Location = new System.Drawing.Point(448, 32);
    this.btnClearHuffman6.Name = "btnClearHuffman6";
    this.btnClearHuffman6.Size = new System.Drawing.Size(40, 16);
    this.btnClearHuffman6.TabIndex = 61;
    this.btnClearHuffman6.Text = "Clear";
    this.btnClearHuffman6.Click += new
                System.EventHandler(this.btnClearHuffman6_Click);
    //
    // btnAddRandomHuffman6
    //
    this.btnAddRandomHuffman6.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
```

```
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnAddRandomHuffman6.Location = new System.Drawing.Point(496, 32);
    this.btnAddRandomHuffman6.Name = "btnAddRandomHuffman6";
    this.btnAddRandomHuffman6.Size = new System.Drawing.Size(48, 16);
    this.btnAddRandomHuffman6.TabIndex = 60;
    this.btnAddRandomHuffman6.Text = "Random";
    this.btnAddRandomHuffman6.Click += new
                System.EventHandler(this.btnAddRandomHuffman6_Click);
    //
    // btnClearHuffman5
    //
    this.btnClearHuffman5.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnClearHuffman5.Location = new System.Drawing.Point(8, 32);
    this.btnClearHuffman5.Name = "btnClearHuffman5";
    this.btnClearHuffman5.Size = new System.Drawing.Size(40, 16);
    this.btnClearHuffman5.TabIndex = 59;
    this.btnClearHuffman5.Text = "Clear";
    this.btnClearHuffman5.Click += new
                System.EventHandler(this.btnClearHuffman5_Click);
    //
    // btnAddRandomHuffman5
    //
    this.btnAddRandomHuffman5.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnAddRandomHuffman5.Location = new System.Drawing.Point(56, 32);
    this.btnAddRandomHuffman5.Name = "btnAddRandomHuffman5";
    this.btnAddRandomHuffman5.Size = new System.Drawing.Size(48, 16);
    this.btnAddRandomHuffman5.TabIndex = 58;
    this.btnAddRandomHuffman5.Text = "Random";
    this.btnAddRandomHuffman5.Click += new
                System.EventHandler(this.btnAddRandomHuffman5_Click);
    //
    // btnRestoreHuffman8
    //
    this.btnRestoreHuffman8.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnRestoreHuffman8.Location = new System.Drawing.Point(496, 208);
    this.btnRestoreHuffman8.Name = "btnRestoreHuffman8";
    this.btnRestoreHuffman8.Size = new System.Drawing.Size(48, 16);
    this.btnRestoreHuffman8.TabIndex = 55;
    this.btnRestoreHuffman8.Text = "Restore";
    this.btnRestoreHuffman8.Click += new
                System.EventHandler(this.btnRestoreHuffman8_Click);
    //
    // btnRestoreHuffman7
    //
    this.btnRestoreHuffman7.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnRestoreHuffman7.Location = new System.Drawing.Point(56, 208);
    this.btnRestoreHuffman7.Name = "btnRestoreHuffman7";
    this.btnRestoreHuffman7.Size = new System.Drawing.Size(48, 16);
    this.btnRestoreHuffman7.TabIndex = 54;
    this.btnRestoreHuffman7.Text = "Restore";
    this.btnRestoreHuffman7.Click += new
                System.EventHandler(this.btnRestoreHuffman7_Click);
    //
    // btnRestoreHuffman6
    //
    this.btnRestoreHuffman6.Font = new
```



```
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnRestoreHuffman6.Location = new System.Drawing.Point(496, 88);
    this.btnRestoreHuffman6.Name = "btnRestoreHuffman6";
    this.btnRestoreHuffman6.Size = new System.Drawing.Size(48, 16);
    this.btnRestoreHuffman6.TabIndex = 53;
    this.btnRestoreHuffman6.Text = "Restore";
    this.btnRestoreHuffman6.Click += new
                System.EventHandler(this.btnRestoreHuffman6_Click);
    //
    // btnRestoreHuffman5
    //
    this.btnRestoreHuffman5.Font = new
                System.Drawing.Font("Microsoft Sans Serif", 7F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.btnRestoreHuffman5.Location = new System.Drawing.Point(56, 88);
    this.btnRestoreHuffman5.Name = "btnRestoreHuffman5";
    this.btnRestoreHuffman5.Size = new System.Drawing.Size(48, 16);
    this.btnRestoreHuffman5.TabIndex = 52;
    this.btnRestoreHuffman5.Text = "Restore";
    this.btnRestoreHuffman5.Click += new
                System.EventHandler(this.btnRestoreHuffman5_Click);
    //
    // txtHuffmanOriginal8
    //
    this.txtHuffmanOriginal8.AutoSize = false;
    this.txtHuffmanOriginal8.Enabled = false;
    this.txtHuffmanOriginal8.Location = new System.Drawing.Point(552, 187);
    this.txtHuffmanOriginal8.Multiline = true;
    this.txtHuffmanOriginal8.Name = "txtHuffmanOriginal8";
    this.txtHuffmanOriginal8.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtHuffmanOriginal8.Size = new System.Drawing.Size(328, 48);
    this.txtHuffmanOriginal8.TabIndex = 50;
    this.txtHuffmanOriginal8.TabStop = false;
    this.txtHuffmanOriginal8.Text = "";
    //
    // lblHuffmanOriginalMarker8
    //
    this.lblHuffmanOriginalMarker8.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblHuffmanOriginalMarker8.Enabled = false;
    this.lblHuffmanOriginalMarker8.Location = new
                System.Drawing.Point(512, 184);
    this.lblHuffmanOriginalMarker8.Name = "lblHuffmanOriginalMarker8";
    this.lblHuffmanOriginalMarker8.Size = new System.Drawing.Size(32, 16);
    this.lblHuffmanOriginalMarker8.TabIndex = 49;
    this.lblHuffmanOriginalMarker8.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblHuffmanOriginal8
    //
    this.lblHuffmanOriginal8.Location = new System.Drawing.Point(456, 184);
    this.lblHuffmanOriginal8.Name = "lblHuffmanOriginal8";
    this.lblHuffmanOriginal8.Size = new System.Drawing.Size(64, 16);
    this.lblHuffmanOriginal8.TabIndex = 48;
    this.lblHuffmanOriginal8.Text = "Original 8:";
    //
    // txtHuffman8
    //
    this.txtHuffman8.AutoSize = false;
    this.txtHuffman8.Location = new System.Drawing.Point(552, 131);
    this.txtHuffman8.Multiline = true;
    this.txtHuffman8.Name = "txtHuffman8";
    this.txtHuffman8.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtHuffman8.Size = new System.Drawing.Size(328, 48);
```

```
    this.txtHuffman8.TabIndex = 32;
    this.txtHuffman8.Text = "";
    this.txtHuffman8.GotFocus += new
                System.EventHandler(this.txtHuffman8_GotFocus);
    //
    // lblHuffmanMarker8
    //
    this.lblHuffmanMarker8.BackColor = System.Drawing.SystemColors.Window;
    this.lblHuffmanMarker8.Enabled = false;
    this.lblHuffmanMarker8.Location = new System.Drawing.Point(512, 128);
    this.lblHuffmanMarker8.Name = "lblHuffmanMarker8";
    this.lblHuffmanMarker8.Size = new System.Drawing.Size(32, 16);
    this.lblHuffmanMarker8.TabIndex = 47;
    this.lblHuffmanMarker8.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblHuffman8
    //
    this.lblHuffman8.Location = new System.Drawing.Point(456, 128);
    this.lblHuffman8.Name = "lblHuffman8";
    this.lblHuffman8.Size = new System.Drawing.Size(64, 16);
    this.lblHuffman8.TabIndex = 46;
    this.lblHuffman8.Text = "Huffman 8:";
    //
    // txtHuffmanOriginal6
    //
    this.txtHuffmanOriginal6.AutoSize = false;
    this.txtHuffmanOriginal6.Enabled = false;
    this.txtHuffmanOriginal6.Location = new System.Drawing.Point(552, 67);
    this.txtHuffmanOriginal6.Multiline = true;
    this.txtHuffmanOriginal6.Name = "txtHuffmanOriginal6";
    this.txtHuffmanOriginal6.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtHuffmanOriginal6.Size = new System.Drawing.Size(328, 48);
    this.txtHuffmanOriginal6.TabIndex = 45;
    this.txtHuffmanOriginal6.TabStop = false;
    this.txtHuffmanOriginal6.Text = "";
    //
    // lblHuffmanOriginalMarker6
    //
    this.lblHuffmanOriginalMarker6.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblHuffmanOriginalMarker6.Enabled = false;
    this.lblHuffmanOriginalMarker6.Location = new
                System.Drawing.Point(512, 64);
    this.lblHuffmanOriginalMarker6.Name = "lblHuffmanOriginalMarker6";
    this.lblHuffmanOriginalMarker6.Size = new System.Drawing.Size(32, 16);
    this.lblHuffmanOriginalMarker6.TabIndex = 44;
    this.lblHuffmanOriginalMarker6.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblHuffmanOriginal6
    //
    this.lblHuffmanOriginal6.Location = new System.Drawing.Point(456, 64);
    this.lblHuffmanOriginal6.Name = "lblHuffmanOriginal6";
    this.lblHuffmanOriginal6.Size = new System.Drawing.Size(64, 16);
    this.lblHuffmanOriginal6.TabIndex = 43;
    this.lblHuffmanOriginal6.Text = "Original 6:";
    //
    // txtHuffman6
    //
    this.txtHuffman6.AutoSize = false;
    this.txtHuffman6.Location = new System.Drawing.Point(552, 11);
    this.txtHuffman6.Multiline = true;
    this.txtHuffman6.Name = "txtHuffman6";
    this.txtHuffman6.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtHuffman6.Size = new System.Drawing.Size(328, 48);
    this.txtHuffman6.TabIndex = 29;
```



```
    this.txtHuffman6.Text = "";
    this.txtHuffman6.GotFocus += new
                System.EventHandler(this.txtHuffman6_GotFocus);
    //
    // lblHuffmanMarker6
    //
    this.lblHuffmanMarker6.BackColor = System.Drawing.SystemColors.Window;
    this.lblHuffmanMarker6.Enabled = false;
    this.lblHuffmanMarker6.Location = new System.Drawing.Point(512, 8);
    this.lblHuffmanMarker6.Name = "lblHuffmanMarker6";
    this.lblHuffmanMarker6.Size = new System.Drawing.Size(32, 16);
    this.lblHuffmanMarker6.TabIndex = 42;
    this.lblHuffmanMarker6.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblHuffman6
    //
    this.lblHuffman6.Location = new System.Drawing.Point(456, 8);
    this.lblHuffman6.Name = "lblHuffman6";
    this.lblHuffman6.Size = new System.Drawing.Size(64, 16);
    this.lblHuffman6.TabIndex = 41;
    this.lblHuffman6.Text = "Huffman 6:";
    //
    // txtHuffmanOriginal7
    //
    this.txtHuffmanOriginal7.AutoSize = false;
    this.txtHuffmanOriginal7.Enabled = false;
    this.txtHuffmanOriginal7.Location = new System.Drawing.Point(112, 187);
    this.txtHuffmanOriginal7.Multiline = true;
    this.txtHuffmanOriginal7.Name = "txtHuffmanOriginal7";
    this.txtHuffmanOriginal7.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtHuffmanOriginal7.Size = new System.Drawing.Size(328, 48);
    this.txtHuffmanOriginal7.TabIndex = 40;
    this.txtHuffmanOriginal7.TabStop = false;
    this.txtHuffmanOriginal7.Text = "";
    //
    // lblHuffmanOriginalMarker7
    //
    this.lblHuffmanOriginalMarker7.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblHuffmanOriginalMarker7.Enabled = false;
    this.lblHuffmanOriginalMarker7.Location = new
                System.Drawing.Point(72, 184);
    this.lblHuffmanOriginalMarker7.Name = "lblHuffmanOriginalMarker7";
    this.lblHuffmanOriginalMarker7.Size = new System.Drawing.Size(32, 16);
    this.lblHuffmanOriginalMarker7.TabIndex = 39;
    this.lblHuffmanOriginalMarker7.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblHuffmanOriginal7
    //
    this.lblHuffmanOriginal7.Location = new System.Drawing.Point(16, 184);
    this.lblHuffmanOriginal7.Name = "lblHuffmanOriginal7";
    this.lblHuffmanOriginal7.Size = new System.Drawing.Size(64, 16);
    this.lblHuffmanOriginal7.TabIndex = 38;
    this.lblHuffmanOriginal7.Text = "Original 7:";
    //
    // txtHuffman7
    //
    this.txtHuffman7.AutoSize = false;
    this.txtHuffman7.Location = new System.Drawing.Point(112, 131);
    this.txtHuffman7.Multiline = true;
    this.txtHuffman7.Name = "txtHuffman7";
    this.txtHuffman7.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtHuffman7.Size = new System.Drawing.Size(328, 48);
    this.txtHuffman7.TabIndex = 31;
    this.txtHuffman7.Text = "";
```

```
            this.txtHuffman7.GotFocus += new
                        System.EventHandler(this.txtHuffman7_GotFocus);
            //
            // lblHuffmanMarker7
            //
            this.lblHuffmanMarker7.BackColor = System.Drawing.SystemColors.Window;
            this.lblHuffmanMarker7.Enabled = false;
            this.lblHuffmanMarker7.Location = new System.Drawing.Point(72, 128);
            this.lblHuffmanMarker7.Name = "lblHuffmanMarker7";
            this.lblHuffmanMarker7.Size = new System.Drawing.Size(32, 16);
            this.lblHuffmanMarker7.TabIndex = 37;
            this.lblHuffmanMarker7.TextAlign =
                        System.Drawing.ContentAlignment.TopCenter;
            //
            // lblHuffman7
            //
            this.lblHuffman7.Location = new System.Drawing.Point(16, 128);
            this.lblHuffman7.Name = "lblHuffman7";
            this.lblHuffman7.Size = new System.Drawing.Size(64, 16);
            this.lblHuffman7.TabIndex = 36;
            this.lblHuffman7.Text = "Huffman 7:";
            //
            // txtHuffmanOriginal5
            //
            this.txtHuffmanOriginal5.AutoSize = false;
            this.txtHuffmanOriginal5.Enabled = false;
            this.txtHuffmanOriginal5.Location = new System.Drawing.Point(112, 67);
            this.txtHuffmanOriginal5.Multiline = true;
            this.txtHuffmanOriginal5.Name = "txtHuffmanOriginal5";
            this.txtHuffmanOriginal5.ScrollBars =
                        System.Windows.Forms.ScrollBars.Horizontal;
            this.txtHuffmanOriginal5.Size = new System.Drawing.Size(328, 48);
            this.txtHuffmanOriginal5.TabIndex = 35;
            this.txtHuffmanOriginal5.TabStop = false;
            this.txtHuffmanOriginal5.Text = "";
            //
            // lblHuffmanOriginalMarker5
            //
            this.lblHuffmanOriginalMarker5.BackColor =
                        System.Drawing.SystemColors.Window;
            this.lblHuffmanOriginalMarker5.Enabled = false;
            this.lblHuffmanOriginalMarker5.Location = new
                        System.Drawing.Point(72, 64);
            this.lblHuffmanOriginalMarker5.Name = "lblHuffmanOriginalMarker5";
            this.lblHuffmanOriginalMarker5.Size = new System.Drawing.Size(32, 16);
            this.lblHuffmanOriginalMarker5.TabIndex = 34;
            this.lblHuffmanOriginalMarker5.TextAlign =
                        System.Drawing.ContentAlignment.TopCenter;
            //
            // lblHuffmanOriginal5
            //
            this.lblHuffmanOriginal5.Location = new System.Drawing.Point(16, 64);
            this.lblHuffmanOriginal5.Name = "lblHuffmanOriginal5";
            this.lblHuffmanOriginal5.Size = new System.Drawing.Size(64, 16);
            this.lblHuffmanOriginal5.TabIndex = 33;
            this.lblHuffmanOriginal5.Text = "Original 5:";
            //
            // txtHuffman5
            //
            this.txtHuffman5.AutoSize = false;
            this.txtHuffman5.Location = new System.Drawing.Point(112, 11);
            this.txtHuffman5.Multiline = true;
            this.txtHuffman5.Name = "txtHuffman5";
            this.txtHuffman5.ScrollBars =
                        System.Windows.Forms.ScrollBars.Horizontal;
            this.txtHuffman5.Size = new System.Drawing.Size(328, 48);
            this.txtHuffman5.TabIndex = 27;
            this.txtHuffman5.Text = "";
            this.txtHuffman5.GotFocus += new
```



```
                        System.EventHandler(this.txtHuffman5_GotFocus);
            //
            // lblHuffmanMarker5
            //
            this.lblHuffmanMarker5.BackColor = System.Drawing.SystemColors.Window;
            this.lblHuffmanMarker5.Enabled = false;
            this.lblHuffmanMarker5.Location = new System.Drawing.Point(72, 8);
            this.lblHuffmanMarker5.Name = "lblHuffmanMarker5";
            this.lblHuffmanMarker5.Size = new System.Drawing.Size(32, 16);
            this.lblHuffmanMarker5.TabIndex = 30;
            this.lblHuffmanMarker5.TextAlign =
                        System.Drawing.ContentAlignment.TopCenter;
            //
            // lblHuffman5
            //
            this.lblHuffman5.Location = new System.Drawing.Point(16, 8);
            this.lblHuffman5.Name = "lblHuffman5";
            this.lblHuffman5.Size = new System.Drawing.Size(64, 16);
            this.lblHuffman5.TabIndex = 28;
            this.lblHuffman5.Text = "Huffman 5:";
            //
            // tabQuantizer
            //
            this.tabQuantizer.Controls.Add(this.txtQuantizerTableNum4);
            this.tabQuantizer.Controls.Add(this.lblQuantizerTableNum4);
            this.tabQuantizer.Controls.Add(this.txtQuantizerTableNum3);
            this.tabQuantizer.Controls.Add(this.lblQuantizerTableNum3);
            this.tabQuantizer.Controls.Add(this.txtQuantizerTableNum2);
            this.tabQuantizer.Controls.Add(this.lblQuantizerTableNum2);
            this.tabQuantizer.Controls.Add(this.txtQuantizerTableNum1);
            this.tabQuantizer.Controls.Add(this.lblQuantizerTableNum1);
            this.tabQuantizer.Controls.Add(this.btnClearQuantizer4);
            this.tabQuantizer.Controls.Add(this.btnAddRandomQuantizer4);
            this.tabQuantizer.Controls.Add(this.btnClearQuantizer3);
            this.tabQuantizer.Controls.Add(this.btnAddRandomQuantizer3);
            this.tabQuantizer.Controls.Add(this.btnClearQuantizer2);
            this.tabQuantizer.Controls.Add(this.btnAddRandomQuantizer2);
            this.tabQuantizer.Controls.Add(this.btnClearQuantizer1);
            this.tabQuantizer.Controls.Add(this.btnAddRandomQuantizer1);
            this.tabQuantizer.Controls.Add(this.btnRestoreQuantizer4);
            this.tabQuantizer.Controls.Add(this.btnRestoreQuantizer3);
            this.tabQuantizer.Controls.Add(this.btnRestoreQuantizer2);
            this.tabQuantizer.Controls.Add(this.btnRestoreQuantizer1);
            this.tabQuantizer.Controls.Add(this.txtQuantizerOriginal4);
            this.tabQuantizer.Controls.Add(this.lblQuantizerOriginalMarker4);
            this.tabQuantizer.Controls.Add(this.lblQuantizerOriginal4);
            this.tabQuantizer.Controls.Add(this.txtQuantizer4);
            this.tabQuantizer.Controls.Add(this.lblQuantizerMarker4);
            this.tabQuantizer.Controls.Add(this.lblQuantizer4);
            this.tabQuantizer.Controls.Add(this.txtQuantizerOriginal2);
            this.tabQuantizer.Controls.Add(this.lblQuantizerOriginalMarker2);
            this.tabQuantizer.Controls.Add(this.lblQuantizerOriginal2);
            this.tabQuantizer.Controls.Add(this.txtQuantizer2);
            this.tabQuantizer.Controls.Add(this.lblQuantizerMarker2);
            this.tabQuantizer.Controls.Add(this.lblQuantizer2);
            this.tabQuantizer.Controls.Add(this.txtQuantizerOriginal3);
            this.tabQuantizer.Controls.Add(this.lblQuantizerOriginalMarker3);
            this.tabQuantizer.Controls.Add(this.lblQuantizerOriginal3);
            this.tabQuantizer.Controls.Add(this.txtQuantizer3);
            this.tabQuantizer.Controls.Add(this.lblQuantizerMarker3);
            this.tabQuantizer.Controls.Add(this.lblQuantizer3);
            this.tabQuantizer.Controls.Add(this.txtQuantizerOriginal1);
            this.tabQuantizer.Controls.Add(this.lblQuantizerOriginalMarker1);
            this.tabQuantizer.Controls.Add(this.lblQuantizerOriginal1);
            this.tabQuantizer.Controls.Add(this.txtQuantizer1);
            this.tabQuantizer.Controls.Add(this.lblQuantizerMarker1);
            this.tabQuantizer.Controls.Add(this.lblQuantizer1);
            this.tabQuantizer.Location = new System.Drawing.Point(4, 22);
            this.tabQuantizer.Name = "tabQuantizer";
```

```
        this.tabQuantizer.Size = new System.Drawing.Size(888, 246);
        this.tabQuantizer.TabIndex = 1;
        this.tabQuantizer.Text = "Quantizer Table";
        //
        // txtQuantizerTableNum4
        //
        this.txtQuantizerTableNum4.BackColor =
                    System.Drawing.SystemColors.Window;
        this.txtQuantizerTableNum4.Enabled = false;
        this.txtQuantizerTableNum4.Location = new
                    System.Drawing.Point(512, 152);
        this.txtQuantizerTableNum4.Name = "txtQuantizerTableNum4";
        this.txtQuantizerTableNum4.Size = new System.Drawing.Size(32, 16);
        this.txtQuantizerTableNum4.TabIndex = 73;
        this.txtQuantizerTableNum4.TextAlign =
                    System.Drawing.ContentAlignment.TopCenter;
        //
        // lblQuantizerTableNum4
        //
        this.lblQuantizerTableNum4.Location = new
                    System.Drawing.Point(448, 152);
        this.lblQuantizerTableNum4.Name = "lblQuantizerTableNum4";
        this.lblQuantizerTableNum4.Size = new System.Drawing.Size(56, 16);
        this.lblQuantizerTableNum4.TabIndex = 72;
        this.lblQuantizerTableNum4.Text = "Table #:";
        //
        // txtQuantizerTableNum3
        //
        this.txtQuantizerTableNum3.BackColor =
                    System.Drawing.SystemColors.Window;
        this.txtQuantizerTableNum3.Enabled = false;
        this.txtQuantizerTableNum3.Location = new System.Drawing.Point(72, 152);
        this.txtQuantizerTableNum3.Name = "txtQuantizerTableNum3";
        this.txtQuantizerTableNum3.Size = new System.Drawing.Size(32, 16);
        this.txtQuantizerTableNum3.TabIndex = 71;
        this.txtQuantizerTableNum3.TextAlign =
                    System.Drawing.ContentAlignment.TopCenter;
        //
        // lblQuantizerTableNum3
        //
        this.lblQuantizerTableNum3.Location = new System.Drawing.Point(8, 152);
        this.lblQuantizerTableNum3.Name = "lblQuantizerTableNum3";
        this.lblQuantizerTableNum3.Size = new System.Drawing.Size(56, 16);
        this.lblQuantizerTableNum3.TabIndex = 70;
        this.lblQuantizerTableNum3.Text = "Table #:";
        //
        // txtQuantizerTableNum2
        //
        this.txtQuantizerTableNum2.BackColor =
                    System.Drawing.SystemColors.Window;
        this.txtQuantizerTableNum2.Enabled = false;
        this.txtQuantizerTableNum2.Location = new
                    System.Drawing.Point(512, 32);
        this.txtQuantizerTableNum2.Name = "txtQuantizerTableNum2";
        this.txtQuantizerTableNum2.Size = new System.Drawing.Size(32, 16);
        this.txtQuantizerTableNum2.TabIndex = 69;
        this.txtQuantizerTableNum2.TextAlign =
                    System.Drawing.ContentAlignment.TopCenter;
        //
        // lblQuantizerTableNum2
        //
        this.lblQuantizerTableNum2.Location = new
                    System.Drawing.Point(448, 32);
        this.lblQuantizerTableNum2.Name = "lblQuantizerTableNum2";
        this.lblQuantizerTableNum2.Size = new System.Drawing.Size(56, 16);
        this.lblQuantizerTableNum2.TabIndex = 68;
        this.lblQuantizerTableNum2.Text = "Table #:";
        //
        // txtQuantizerTableNum1
```



```
        //
        this.txtQuantizerTableNum1.BackColor =
                    System.Drawing.SystemColors.Window;
        this.txtQuantizerTableNum1.Enabled = false;
        this.txtQuantizerTableNum1.Location = new System.Drawing.Point(72, 32);
        this.txtQuantizerTableNum1.Name = "txtQuantizerTableNum1";
        this.txtQuantizerTableNum1.Size = new System.Drawing.Size(32, 16);
        this.txtQuantizerTableNum1.TabIndex = 67;
        this.txtQuantizerTableNum1.TextAlign =
                    System.Drawing.ContentAlignment.TopCenter;
        //
        // lblQuantizerTableNum1
        //
        this.lblQuantizerTableNum1.Location = new System.Drawing.Point(8, 32);
        this.lblQuantizerTableNum1.Name = "lblQuantizerTableNum1";
        this.lblQuantizerTableNum1.Size = new System.Drawing.Size(56, 16);
        this.lblQuantizerTableNum1.TabIndex = 66;
        this.lblQuantizerTableNum1.Text = "Table #:";
        //
        // btnClearQuantizer4
        //
        this.btnClearQuantizer4.Font = new
                    System.Drawing.Font("Microsoft Sans Serif", 7F,
                        System.Drawing.FontStyle.Regular,
                        System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.btnClearQuantizer4.Location = new System.Drawing.Point(448, 176);
        this.btnClearQuantizer4.Name = "btnClearQuantizer4";
        this.btnClearQuantizer4.Size = new System.Drawing.Size(40, 16);
        this.btnClearQuantizer4.TabIndex = 65;
        this.btnClearQuantizer4.Text = "Clear";
        this.btnClearQuantizer4.Click += new
                    System.EventHandler(this.btnClearQuantizer4_Click);
        //
        // btnAddRandomQuantizer4
        //
        this.btnAddRandomQuantizer4.Font = new
                    System.Drawing.Font("Microsoft Sans Serif", 7F,
                        System.Drawing.FontStyle.Regular,
                        System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.btnAddRandomQuantizer4.Location = new
                    System.Drawing.Point(496, 176);
        this.btnAddRandomQuantizer4.Name = "btnAddRandomQuantizer4";
        this.btnAddRandomQuantizer4.Size = new System.Drawing.Size(48, 16);
        this.btnAddRandomQuantizer4.TabIndex = 64;
        this.btnAddRandomQuantizer4.Text = "Random";
        this.btnAddRandomQuantizer4.Click += new
                    System.EventHandler(this.btnAddRandomQuantizer4_Click);
        //
        // btnClearQuantizer3
        //
        this.btnClearQuantizer3.Font = new
                    System.Drawing.Font("Microsoft Sans Serif", 7F,
                        System.Drawing.FontStyle.Regular,
                        System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.btnClearQuantizer3.Location = new System.Drawing.Point(8, 176);
        this.btnClearQuantizer3.Name = "btnClearQuantizer3";
        this.btnClearQuantizer3.Size = new System.Drawing.Size(40, 16);
        this.btnClearQuantizer3.TabIndex = 63;
        this.btnClearQuantizer3.Text = "Clear";
        this.btnClearQuantizer3.Click += new
                    System.EventHandler(this.btnClearQuantizer3_Click);
        //
        // btnAddRandomQuantizer3
        //
        this.btnAddRandomQuantizer3.Font = new
                    System.Drawing.Font("Microsoft Sans Serif", 7F,
                        System.Drawing.FontStyle.Regular,
                        System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.btnAddRandomQuantizer3.Location = new
```

```
            System.Drawing.Point(56, 176);
this.btnAddRandomQuantizer3.Name = "btnAddRandomQuantizer3";
this.btnAddRandomQuantizer3.Size = new System.Drawing.Size(48, 16);
this.btnAddRandomQuantizer3.TabIndex = 62;
this.btnAddRandomQuantizer3.Text = "Random";
this.btnAddRandomQuantizer3.Click += new
            System.EventHandler(this.btnAddRandomQuantizer3_Click);
//
// btnClearQuantizer2
//
this.btnClearQuantizer2.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnClearQuantizer2.Location = new System.Drawing.Point(448, 56);
this.btnClearQuantizer2.Name = "btnClearQuantizer2";
this.btnClearQuantizer2.Size = new System.Drawing.Size(40, 16);
this.btnClearQuantizer2.TabIndex = 61;
this.btnClearQuantizer2.Text = "Clear";
this.btnClearQuantizer2.Click += new
            System.EventHandler(this.btnClearQuantizer2_Click);
//
// btnAddRandomQuantizer2
//
this.btnAddRandomQuantizer2.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                 System.Drawing.FontStyle.Regular,
                 System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnAddRandomQuantizer2.Location = new
            System.Drawing.Point(496, 56);
this.btnAddRandomQuantizer2.Name = "btnAddRandomQuantizer2";
this.btnAddRandomQuantizer2.Size = new System.Drawing.Size(48, 16);
this.btnAddRandomQuantizer2.TabIndex = 60;
this.btnAddRandomQuantizer2.Text = "Random";
this.btnAddRandomQuantizer2.Click += new
            System.EventHandler(this.btnAddRandomQuantizer2_Click);
//
// btnClearQuantizer1
//
this.btnClearQuantizer1.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnClearQuantizer1.Location = new System.Drawing.Point(8, 56);
this.btnClearQuantizer1.Name = "btnClearQuantizer1";
this.btnClearQuantizer1.Size = new System.Drawing.Size(40, 16);
this.btnClearQuantizer1.TabIndex = 59;
this.btnClearQuantizer1.Text = "Clear";
this.btnClearQuantizer1.Click += new
            System.EventHandler(this.btnClearQuantizer1_Click);
//
// btnAddRandomQuantizer1
//
this.btnAddRandomQuantizer1.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnAddRandomQuantizer1.Location = new
            System.Drawing.Point(56, 56);
this.btnAddRandomQuantizer1.Name = "btnAddRandomQuantizer1";
this.btnAddRandomQuantizer1.Size = new System.Drawing.Size(48, 16);
this.btnAddRandomQuantizer1.TabIndex = 58;
this.btnAddRandomQuantizer1.Text = "Random";
this.btnAddRandomQuantizer1.Click += new
            System.EventHandler(this.btnAddRandomQuantizer1_Click);
//
// btnRestoreQuantizer4
//
this.btnRestoreQuantizer4.Font = new
```



```
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnRestoreQuantizer4.Location = new
            System.Drawing.Point(496, 224);
this.btnRestoreQuantizer4.Name = "btnRestoreQuantizer4";
this.btnRestoreQuantizer4.Size = new System.Drawing.Size(48, 16);
this.btnRestoreQuantizer4.TabIndex = 54;
this.btnRestoreQuantizer4.Text = "Restore";
this.btnRestoreQuantizer4.Click += new
            System.EventHandler(this.btnRestoreQuantizer4_Click);
//
// btnRestoreQuantizer3
//
this.btnRestoreQuantizer3.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnRestoreQuantizer3.Location = new System.Drawing.Point(56, 224);
this.btnRestoreQuantizer3.Name = "btnRestoreQuantizer3";
this.btnRestoreQuantizer3.Size = new System.Drawing.Size(48, 16);
this.btnRestoreQuantizer3.TabIndex = 53;
this.btnRestoreQuantizer3.Text = "Restore";
this.btnRestoreQuantizer3.Click += new
            System.EventHandler(this.btnRestoreQuantizer3_Click);
//
// btnRestoreQuantizer2
//
this.btnRestoreQuantizer2.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                 System.Drawing.FontStyle.Regular,
                 System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnRestoreQuantizer2.Location = new System.Drawing.Point(496, 104);
this.btnRestoreQuantizer2.Name = "btnRestoreQuantizer2";
this.btnRestoreQuantizer2.Size = new System.Drawing.Size(48, 16);
this.btnRestoreQuantizer2.TabIndex = 52;
this.btnRestoreQuantizer2.Text = "Restore";
this.btnRestoreQuantizer2.Click += new
            System.EventHandler(this.btnRestoreQuantizer2_Click);
//
// btnRestoreQuantizer1
//
this.btnRestoreQuantizer1.Font = new
            System.Drawing.Font("Microsoft Sans Serif", 7F,
                 System.Drawing.FontStyle.Regular,
                 System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.btnRestoreQuantizer1.Location = new System.Drawing.Point(56, 104);
this.btnRestoreQuantizer1.Name = "btnRestoreQuantizer1";
this.btnRestoreQuantizer1.Size = new System.Drawing.Size(48, 16);
this.btnRestoreQuantizer1.TabIndex = 51;
this.btnRestoreQuantizer1.Text = "Restore";
this.btnRestoreQuantizer1.Click += new
            System.EventHandler(this.btnRestoreQuantizer1_Click);
//
// txtQuantizerOriginal4
//
this.txtQuantizerOriginal4.AutoSize = false;
this.txtQuantizerOriginal4.Enabled = false;
this.txtQuantizerOriginal4.Location = new
            System.Drawing.Point(552, 187);
this.txtQuantizerOriginal4.Multiline = true;
this.txtQuantizerOriginal4.Name = "txtQuantizerOriginal4";
this.txtQuantizerOriginal4.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
this.txtQuantizerOriginal4.Size = new System.Drawing.Size(328, 48);
this.txtQuantizerOriginal4.TabIndex = 50;
this.txtQuantizerOriginal4.TabStop = false;
this.txtQuantizerOriginal4.Text = "";
//
```

```
    // lblQuantizerOriginalMarker4
    //
    this.lblQuantizerOriginalMarker4.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblQuantizerOriginalMarker4.Enabled = false;
    this.lblQuantizerOriginalMarker4.Location = new
                System.Drawing.Point(512, 200);
    this.lblQuantizerOriginalMarker4.Name = "lblQuantizerOriginalMarker4";
    this.lblQuantizerOriginalMarker4.Size = new
                System.Drawing.Size(32, 16);
    this.lblQuantizerOriginalMarker4.TabIndex = 49;
    this.lblQuantizerOriginalMarker4.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblQuantizerOriginal4
    //
    this.lblQuantizerOriginal4.Location = new
                System.Drawing.Point(448, 200);
    this.lblQuantizerOriginal4.Name = "lblQuantizerOriginal4";
    this.lblQuantizerOriginal4.Size = new System.Drawing.Size(72, 16);
    this.lblQuantizerOriginal4.TabIndex = 48;
    this.lblQuantizerOriginal4.Text = "Original 4:";
    //
    // txtQuantizer4
    //
    this.txtQuantizer4.AutoSize = false;
    this.txtQuantizer4.Location = new System.Drawing.Point(552, 131);
    this.txtQuantizer4.Multiline = true;
    this.txtQuantizer4.Name = "txtQuantizer4";
    this.txtQuantizer4.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtQuantizer4.Size = new System.Drawing.Size(328, 48);
    this.txtQuantizer4.TabIndex = 3;
    this.txtQuantizer4.Text = "";
    this.txtQuantizer4.GotFocus += new
                System.EventHandler(this.txtQuantizer4_Click);
    this.txtQuantizer4.Click += new
                System.EventHandler(this.txtQuantizer4_Click);
    //
    // lblQuantizerMarker4
    //
    this.lblQuantizerMarker4.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblQuantizerMarker4.Enabled = false;
    this.lblQuantizerMarker4.Location = new
                System.Drawing.Point(512, 128);
    this.lblQuantizerMarker4.Name = "lblQuantizerMarker4";
    this.lblQuantizerMarker4.Size = new System.Drawing.Size(32, 16);
    this.lblQuantizerMarker4.TabIndex = 46;
    this.lblQuantizerMarker4.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblQuantizer4
    //
    this.lblQuantizer4.Location = new System.Drawing.Point(448, 128);
    this.lblQuantizer4.Name = "lblQuantizer4";
    this.lblQuantizer4.Size = new System.Drawing.Size(72, 16);
    this.lblQuantizer4.TabIndex = 45;
    this.lblQuantizer4.Text = "Quantizer 4:";
    //
    // txtQuantizerOriginal2
    //
    this.txtQuantizerOriginal2.AutoSize = false;
    this.txtQuantizerOriginal2.Enabled = false;
    this.txtQuantizerOriginal2.Location = new
                System.Drawing.Point(552, 67);
    this.txtQuantizerOriginal2.Multiline = true;
    this.txtQuantizerOriginal2.Name = "txtQuantizerOriginal2";
    this.txtQuantizerOriginal2.ScrollBars =
```



```
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtQuantizerOriginal2.Size = new System.Drawing.Size(328, 48);
    this.txtQuantizerOriginal2.TabIndex = 44;
    this.txtQuantizerOriginal2.TabStop = false;
    this.txtQuantizerOriginal2.Text = "";
    //
    // lblQuantizerOriginalMarker2
    //
    this.lblQuantizerOriginalMarker2.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblQuantizerOriginalMarker2.Enabled = false;
    this.lblQuantizerOriginalMarker2.Location = new
                System.Drawing.Point(512, 80);
    this.lblQuantizerOriginalMarker2.Name = "lblQuantizerOriginalMarker2";
    this.lblQuantizerOriginalMarker2.Size = new
                System.Drawing.Size(32, 16);
    this.lblQuantizerOriginalMarker2.TabIndex = 43;
    this.lblQuantizerOriginalMarker2.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblQuantizerOriginal2
    //
    this.lblQuantizerOriginal2.Location = new
                System.Drawing.Point(448, 80);
    this.lblQuantizerOriginal2.Name = "lblQuantizerOriginal2";
    this.lblQuantizerOriginal2.Size = new System.Drawing.Size(72, 16);
    this.lblQuantizerOriginal2.TabIndex = 42;
    this.lblQuantizerOriginal2.Text = "Original 2:";
    //
    // txtQuantizer2
    //
    this.txtQuantizer2.AutoSize = false;
    this.txtQuantizer2.Location = new System.Drawing.Point(552, 11);
    this.txtQuantizer2.Multiline = true;
    this.txtQuantizer2.Name = "txtQuantizer2";
    this.txtQuantizer2.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtQuantizer2.Size = new System.Drawing.Size(328, 48);
    this.txtQuantizer2.TabIndex = 1;
    this.txtQuantizer2.Text = "";
    this.txtQuantizer2.GotFocus += new
                System.EventHandler(this.txtQuantizer2_Click);
    this.txtQuantizer2.Click += new
                System.EventHandler(this.txtQuantizer2_Click);
    //
    // lblQuantizerMarker2
    //
    this.lblQuantizerMarker2.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblQuantizerMarker2.Enabled = false;
    this.lblQuantizerMarker2.Location = new
                System.Drawing.Point(512, 8);
    this.lblQuantizerMarker2.Name = "lblQuantizerMarker2";
    this.lblQuantizerMarker2.Size = new System.Drawing.Size(32, 16);
    this.lblQuantizerMarker2.TabIndex = 40;
    this.lblQuantizerMarker2.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblQuantizer2
    //
    this.lblQuantizer2.Location = new System.Drawing.Point(448, 8);
    this.lblQuantizer2.Name = "lblQuantizer2";
    this.lblQuantizer2.Size = new System.Drawing.Size(72, 16);
    this.lblQuantizer2.TabIndex = 39;
    this.lblQuantizer2.Text = "Quantizer 2:";
    //
    // txtQuantizerOriginal3
    //
    this.txtQuantizerOriginal3.AutoSize = false;
```

```
    this.txtQuantizerOriginal3.Enabled = false;
    this.txtQuantizerOriginal3.Location = new
            System.Drawing.Point(112, 187);
    this.txtQuantizerOriginal3.Multiline = true;
    this.txtQuantizerOriginal3.Name = "txtQuantizerOriginal3";
    this.txtQuantizerOriginal3.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
    this.txtQuantizerOriginal3.Size = new System.Drawing.Size(328, 48);
    this.txtQuantizerOriginal3.TabIndex = 38;
    this.txtQuantizerOriginal3.TabStop = false;
    this.txtQuantizerOriginal3.Text = "";
    //
    // lblQuantizerOriginalMarker3
    //
    this.lblQuantizerOriginalMarker3.BackColor =
            System.Drawing.SystemColors.Window;
    this.lblQuantizerOriginalMarker3.Enabled = false;
    this.lblQuantizerOriginalMarker3.Location = new
            System.Drawing.Point(72, 200);
    this.lblQuantizerOriginalMarker3.Name = "lblQuantizerOriginalMarker3";
    this.lblQuantizerOriginalMarker3.Size = new
            System.Drawing.Size(32, 16);
    this.lblQuantizerOriginalMarker3.TabIndex = 37;
    this.lblQuantizerOriginalMarker3.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
    //
    // lblQuantizerOriginal3
    //
    this.lblQuantizerOriginal3.Location = new System.Drawing.Point(8, 200);
    this.lblQuantizerOriginal3.Name = "lblQuantizerOriginal3";
    this.lblQuantizerOriginal3.Size = new System.Drawing.Size(72, 16);
    this.lblQuantizerOriginal3.TabIndex = 36;
    this.lblQuantizerOriginal3.Text = "Original 3:";
    //
    // txtQuantizer3
    //
    this.txtQuantizer3.AutoSize = false;
    this.txtQuantizer3.Location = new System.Drawing.Point(112, 131);
    this.txtQuantizer3.Multiline = true;
    this.txtQuantizer3.Name = "txtQuantizer3";
    this.txtQuantizer3.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
    this.txtQuantizer3.Size = new System.Drawing.Size(328, 48);
    this.txtQuantizer3.TabIndex = 2;
    this.txtQuantizer3.Text = "";
    this.txtQuantizer3.GotFocus += new
            System.EventHandler(this.txtQuantizer3_Click);
    this.txtQuantizer3.Click += new
            System.EventHandler(this.txtQuantizer3_Click);
    //
    // lblQuantizerMarker3
    //
    this.lblQuantizerMarker3.BackColor =
            System.Drawing.SystemColors.Window;
    this.lblQuantizerMarker3.Enabled = false;
    this.lblQuantizerMarker3.Location = new System.Drawing.Point(72, 128);
    this.lblQuantizerMarker3.Name = "lblQuantizerMarker3";
    this.lblQuantizerMarker3.Size = new System.Drawing.Size(32, 16);
    this.lblQuantizerMarker3.TabIndex = 34;
    this.lblQuantizerMarker3.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
    //
    // lblQuantizer3
    //
    this.lblQuantizer3.Location = new System.Drawing.Point(8, 128);
    this.lblQuantizer3.Name = "lblQuantizer3";
    this.lblQuantizer3.Size = new System.Drawing.Size(72, 16);
    this.lblQuantizer3.TabIndex = 33;
    this.lblQuantizer3.Text = "Quantizer 3:";
```



```
    //
    // txtQuantizerOriginal1
    //
    this.txtQuantizerOriginal1.AutoSize = false;
    this.txtQuantizerOriginal1.Enabled = false;
    this.txtQuantizerOriginal1.Location = new
            System.Drawing.Point(112, 67);
    this.txtQuantizerOriginal1.Multiline = true;
    this.txtQuantizerOriginal1.Name = "txtQuantizerOriginal1";
    this.txtQuantizerOriginal1.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
    this.txtQuantizerOriginal1.Size = new System.Drawing.Size(328, 48);
    this.txtQuantizerOriginal1.TabIndex = 32;
    this.txtQuantizerOriginal1.TabStop = false;
    this.txtQuantizerOriginal1.Text = "";
    //
    // lblQuantizerOriginalMarker1
    //
    this.lblQuantizerOriginalMarker1.BackColor =
            System.Drawing.SystemColors.Window;
    this.lblQuantizerOriginalMarker1.Enabled = false;
    this.lblQuantizerOriginalMarker1.Location = new
            System.Drawing.Point(72, 80);
    this.lblQuantizerOriginalMarker1.Name = "lblQuantizerOriginalMarker1";
    this.lblQuantizerOriginalMarker1.Size = new
            System.Drawing.Size(32, 16);
    this.lblQuantizerOriginalMarker1.TabIndex = 31;
    this.lblQuantizerOriginalMarker1.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
    //
    // lblQuantizerOriginal1
    //
    this.lblQuantizerOriginal1.Location = new System.Drawing.Point(8, 80);
    this.lblQuantizerOriginal1.Name = "lblQuantizerOriginal1";
    this.lblQuantizerOriginal1.Size = new System.Drawing.Size(72, 16);
    this.lblQuantizerOriginal1.TabIndex = 30;
    this.lblQuantizerOriginal1.Text = "Original 1:";
    //
    // txtQuantizer1
    //
    this.txtQuantizer1.AutoSize = false;
    this.txtQuantizer1.Location = new System.Drawing.Point(112, 11);
    this.txtQuantizer1.Multiline = true;
    this.txtQuantizer1.Name = "txtQuantizer1";
    this.txtQuantizer1.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
    this.txtQuantizer1.Size = new System.Drawing.Size(328, 48);
    this.txtQuantizer1.TabIndex = 0;
    this.txtQuantizer1.Text = "";
    this.txtQuantizer1.GotFocus += new
            System.EventHandler(this.txtQuantizer1_Click);
    this.txtQuantizer1.Click += new
            System.EventHandler(this.txtQuantizer1_Click);
    //
    // lblQuantizerMarker1
    //
    this.lblQuantizerMarker1.BackColor =
            System.Drawing.SystemColors.Window;
    this.lblQuantizerMarker1.Enabled = false;
    this.lblQuantizerMarker1.Location = new System.Drawing.Point(72, 8);
    this.lblQuantizerMarker1.Name = "lblQuantizerMarker1";
    this.lblQuantizerMarker1.Size = new System.Drawing.Size(32, 16);
    this.lblQuantizerMarker1.TabIndex = 28;
    this.lblQuantizerMarker1.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
    //
    // lblQuantizer1
    //
    this.lblQuantizer1.Location = new System.Drawing.Point(8, 8);
```

```
      this.lblQuantizer1.Name = "lblQuantizer1";
      this.lblQuantizer1.Size = new System.Drawing.Size(72, 16);
      this.lblQuantizer1.TabIndex = 27;
      this.lblQuantizer1.Text = "Quantizer 1:";
      //
      // tabEncodedData
      //
      this.tabEncodedData.Controls.Add(this.lblOriginalHeader);
      this.tabEncodedData.Controls.Add(this.txtOriginalHeader);
      this.tabEncodedData.Controls.Add(this.lblScanHeader);
      this.tabEncodedData.Controls.Add(this.txtScanHeader);
      this.tabEncodedData.Controls.Add(this.txtOriginalEncodedData);
      this.tabEncodedData.Controls.Add(this.lblOriginalEncodedData);
      this.tabEncodedData.Controls.Add(this.txtEncodedData);
      this.tabEncodedData.Controls.Add(this.lblEncodedData);
      this.tabEncodedData.Location = new System.Drawing.Point(4, 22);
      this.tabEncodedData.Name = "tabEncodedData";
      this.tabEncodedData.Size = new System.Drawing.Size(888, 246);
      this.tabEncodedData.TabIndex = 2;
      this.tabEncodedData.Text = "Encoded Data";
      //
      // lblOriginalHeader
      //
      this.lblOriginalHeader.Location = new System.Drawing.Point(312, 112);
      this.lblOriginalHeader.Name = "lblOriginalHeader";
      this.lblOriginalHeader.Size = new System.Drawing.Size(88, 16);
      this.lblOriginalHeader.TabIndex = 14;
      this.lblOriginalHeader.Text = "Original Header:";
      //
      // txtOriginalHeader
      //
      this.txtOriginalHeader.Enabled = false;
      this.txtOriginalHeader.Location = new System.Drawing.Point(408, 112);
      this.txtOriginalHeader.Name = "txtOriginalHeader";
      this.txtOriginalHeader.Size = new System.Drawing.Size(464, 20);
      this.txtOriginalHeader.TabIndex = 13;
      this.txtOriginalHeader.TabStop = false;
      this.txtOriginalHeader.Text = "";
      this.toolTips.SetToolTip(this.txtOriginalHeader,
            "This is the original Scan Header for the encoded data.");
      //
      // lblScanHeader
      //
      this.lblScanHeader.Location = new System.Drawing.Point(320, 8);
      this.lblScanHeader.Name = "lblScanHeader";
      this.lblScanHeader.Size = new System.Drawing.Size(80, 16);
      this.lblScanHeader.TabIndex = 12;
      this.lblScanHeader.Text = "Scan Header:";
      //
      // txtScanHeader
      //
      this.txtScanHeader.Location = new System.Drawing.Point(408, 8);
      this.txtScanHeader.Name = "txtScanHeader";
      this.txtScanHeader.Size = new System.Drawing.Size(464, 20);
      this.txtScanHeader.TabIndex = 1;
      this.txtScanHeader.Text = "";
      this.toolTips.SetToolTip(this.txtScanHeader,
            "This is the Scan Header desrcibing this particular "+
                  "encoded stream.");
      //
      // txtOriginalEncodedData
      //
      this.txtOriginalEncodedData.Enabled = false;
      this.txtOriginalEncodedData.Location = new
                  System.Drawing.Point(8, 136);
      this.txtOriginalEncodedData.MaxLength = 10240;
      this.txtOriginalEncodedData.Multiline = true;
      this.txtOriginalEncodedData.Name = "txtOriginalEncodedData";
      this.txtOriginalEncodedData.ScrollBars =
```



```
                  System.Windows.Forms.ScrollBars.Horizontal;
      this.txtOriginalEncodedData.Size = new System.Drawing.Size(864, 64);
      this.txtOriginalEncodedData.TabIndex = 10;
      this.txtOriginalEncodedData.TabStop = false;
      this.txtOriginalEncodedData.Text = "";
      this.toolTips.SetToolTip(this.txtOriginalEncodedData,
            "This is the original entropy encoded data stream.");
      //
      // lblOriginalEncodedData
      //
      this.lblOriginalEncodedData.Location = new
                  System.Drawing.Point(8, 120);
      this.lblOriginalEncodedData.Name = "lblOriginalEncodedData";
      this.lblOriginalEncodedData.Size = new System.Drawing.Size(128, 16);
      this.lblOriginalEncodedData.TabIndex = 9;
      this.lblOriginalEncodedData.Text = "Original Encoded Data:";
      //
      // txtEncodedData
      //
      this.txtEncodedData.Location = new System.Drawing.Point(8, 32);
      this.txtEncodedData.MaxLength = 10240;
      this.txtEncodedData.Multiline = true;
      this.txtEncodedData.Name = "txtEncodedData";
      this.txtEncodedData.ScrollBars =
                  System.Windows.Forms.ScrollBars.Horizontal;
      this.txtEncodedData.Size = new System.Drawing.Size(864, 64);
      this.txtEncodedData.TabIndex = 0;
      this.txtEncodedData.Text = "";
      this.toolTips.SetToolTip(this.txtEncodedData,
            "This is the entropy encoded data stream.");
      //
      // lblEncodedData
      //
      this.lblEncodedData.Location = new System.Drawing.Point(8, 16);
      this.lblEncodedData.Name = "lblEncodedData";
      this.lblEncodedData.Size = new System.Drawing.Size(248, 16);
      this.lblEncodedData.TabIndex = 6;
      this.lblEncodedData.Text = "Encoded Data:";
      //
      // tabApplicationData
      //
      this.tabApplicationData.Controls.Add(this.txtApplicationData10);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker10);
      this.tabApplicationData.Controls.Add(this.lblApplicationData10);
      this.tabApplicationData.Controls.Add(this.txtApplicationData9);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker9);
      this.tabApplicationData.Controls.Add(this.lblApplicationData9);
      this.tabApplicationData.Controls.Add(this.txtApplicationData8);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker8);
      this.tabApplicationData.Controls.Add(this.lblApplicationData8);
      this.tabApplicationData.Controls.Add(this.txtApplicationData7);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker7);
      this.tabApplicationData.Controls.Add(this.lblApplicationData7);
      this.tabApplicationData.Controls.Add(this.txtApplicationData6);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker6);
      this.tabApplicationData.Controls.Add(this.lblApplicationData6);
      this.tabApplicationData.Controls.Add(this.txtApplicationData5);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker5);
      this.tabApplicationData.Controls.Add(this.lblApplicationData5);
      this.tabApplicationData.Controls.Add(this.txtApplicationData4);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker4);
      this.tabApplicationData.Controls.Add(this.lblApplicationData4);
      this.tabApplicationData.Controls.Add(this.txtApplicationData3);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker3);
      this.tabApplicationData.Controls.Add(this.lblApplicationData3);
      this.tabApplicationData.Controls.Add(this.txtApplicationData2);
      this.tabApplicationData.Controls.Add(this.lblApplicationMarker2);
      this.tabApplicationData.Controls.Add(this.lblApplicationData2);
      this.tabApplicationData.Controls.Add(this.txtApplicationData1);
```

```
    this.tabApplicationData.Controls.Add(this.lblApplicationMarker1);
    this.tabApplicationData.Controls.Add(this.lblApplicationData1);
    this.tabApplicationData.Location = new System.Drawing.Point(4, 22);
    this.tabApplicationData.Name = "tabApplicationData";
    this.tabApplicationData.Size = new System.Drawing.Size(888, 246);
    this.tabApplicationData.TabIndex = 6;
    this.tabApplicationData.Text = "Application Data";
    //
    // txtApplicationData10
    //
    this.txtApplicationData10.AutoSize = false;
    this.txtApplicationData10.Location = new
                System.Drawing.Point(552, 200);
    this.txtApplicationData10.Multiline = true;
    this.txtApplicationData10.Name = "txtApplicationData10";
    this.txtApplicationData10.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData10.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData10.TabIndex = 9;
    this.txtApplicationData10.Text = "";
    //
    // lblApplicationMarker10
    //
    this.lblApplicationMarker10.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblApplicationMarker10.Enabled = false;
    this.lblApplicationMarker10.Location = new
                System.Drawing.Point(512, 208);
    this.lblApplicationMarker10.Name = "lblApplicationMarker10";
    this.lblApplicationMarker10.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker10.TabIndex = 64;
    this.lblApplicationMarker10.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData10
    //
    this.lblApplicationData10.Location = new
                System.Drawing.Point(440, 208);
    this.lblApplicationData10.Name = "lblApplicationData10";
    this.lblApplicationData10.Size = new System.Drawing.Size(72, 16);
    this.lblApplicationData10.TabIndex = 63;
    this.lblApplicationData10.Text = "App Data 10:";
    //
    // txtApplicationData9
    //
    this.txtApplicationData9.AutoSize = false;
    this.txtApplicationData9.Location = new System.Drawing.Point(104, 200);
    this.txtApplicationData9.Multiline = true;
    this.txtApplicationData9.Name = "txtApplicationData9";
    this.txtApplicationData9.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData9.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData9.TabIndex = 8;
    this.txtApplicationData9.Text = "";
    //
    // lblApplicationMarker9
    //
    this.lblApplicationMarker9.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblApplicationMarker9.Enabled = false;
    this.lblApplicationMarker9.Location = new
                System.Drawing.Point(64, 208);
    this.lblApplicationMarker9.Name = "lblApplicationMarker9";
    this.lblApplicationMarker9.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker9.TabIndex = 61;
    this.lblApplicationMarker9.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData9
```



```
    //
    this.lblApplicationData9.Location = new System.Drawing.Point(0, 208);
    this.lblApplicationData9.Name = "lblApplicationData9";
    this.lblApplicationData9.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData9.TabIndex = 60;
    this.lblApplicationData9.Text = "App Data 9:";
    //
    // txtApplicationData8
    //
    this.txtApplicationData8.AutoSize = false;
    this.txtApplicationData8.Location = new
                System.Drawing.Point(552, 152);
    this.txtApplicationData8.Multiline = true;
    this.txtApplicationData8.Name = "txtApplicationData8";
    this.txtApplicationData8.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData8.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData8.TabIndex = 7;
    this.txtApplicationData8.Text = "";
    //
    // lblApplicationMarker8
    //
    this.lblApplicationMarker8.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblApplicationMarker8.Enabled = false;
    this.lblApplicationMarker8.Location = new
                System.Drawing.Point(512, 160);
    this.lblApplicationMarker8.Name = "lblApplicationMarker8";
    this.lblApplicationMarker8.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker8.TabIndex = 58;
    this.lblApplicationMarker8.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData8
    //
    this.lblApplicationData8.Location = new System.Drawing.Point(448, 160);
    this.lblApplicationData8.Name = "lblApplicationData8";
    this.lblApplicationData8.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData8.TabIndex = 57;
    this.lblApplicationData8.Text = "App Data 8:";
    //
    // txtApplicationData7
    //
    this.txtApplicationData7.AutoSize = false;
    this.txtApplicationData7.Location = new System.Drawing.Point(104, 152);
    this.txtApplicationData7.Multiline = true;
    this.txtApplicationData7.Name = "txtApplicationData7";
    this.txtApplicationData7.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData7.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData7.TabIndex = 6;
    this.txtApplicationData7.Text = "";
    //
    // lblApplicationMarker7
    //
    this.lblApplicationMarker7.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblApplicationMarker7.Enabled = false;
    this.lblApplicationMarker7.Location = new
                System.Drawing.Point(64, 160);
    this.lblApplicationMarker7.Name = "lblApplicationMarker7";
    this.lblApplicationMarker7.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker7.TabIndex = 55;
    this.lblApplicationMarker7.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData7
    //
    this.lblApplicationData7.Location = new System.Drawing.Point(0, 160);
```

```
    this.lblApplicationData7.Name = "lblApplicationData7";
    this.lblApplicationData7.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData7.TabIndex = 54;
    this.lblApplicationData7.Text = "App Data 7:";
    //
    // txtApplicationData6
    //
    this.txtApplicationData6.AutoSize = false;
    this.txtApplicationData6.Location = new System.Drawing.Point(552, 105);
    this.txtApplicationData6.Multiline = true;
    this.txtApplicationData6.Name = "txtApplicationData6";
    this.txtApplicationData6.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData6.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData6.TabIndex = 5;
    this.txtApplicationData6.Text = "";
    //
    // lblApplicationMarker6
    //
    this.lblApplicationMarker6.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblApplicationMarker6.Enabled = false;
    this.lblApplicationMarker6.Location = new
                System.Drawing.Point(512, 112);
    this.lblApplicationMarker6.Name = "lblApplicationMarker6";
    this.lblApplicationMarker6.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker6.TabIndex = 52;
    this.lblApplicationMarker6.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData6
    //
    this.lblApplicationData6.Location = new System.Drawing.Point(448, 112);
    this.lblApplicationData6.Name = "lblApplicationData6";
    this.lblApplicationData6.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData6.TabIndex = 51;
    this.lblApplicationData6.Text = "App Data 6:";
    //
    // txtApplicationData5
    //
    this.txtApplicationData5.AutoSize = false;
    this.txtApplicationData5.Location = new System.Drawing.Point(104, 105);
    this.txtApplicationData5.Multiline = true;
    this.txtApplicationData5.Name = "txtApplicationData5";
    this.txtApplicationData5.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData5.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData5.TabIndex = 4;
    this.txtApplicationData5.Text = "";
    //
    // lblApplicationMarker5
    //
    this.lblApplicationMarker5.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblApplicationMarker5.Enabled = false;
    this.lblApplicationMarker5.Location = new
                System.Drawing.Point(64, 112);
    this.lblApplicationMarker5.Name = "lblApplicationMarker5";
    this.lblApplicationMarker5.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker5.TabIndex = 49;
    this.lblApplicationMarker5.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData5
    //
    this.lblApplicationData5.Location = new System.Drawing.Point(0, 112);
    this.lblApplicationData5.Name = "lblApplicationData5";
    this.lblApplicationData5.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData5.TabIndex = 48;
```



```
    this.lblApplicationData5.Text = "App Data 5:";
    //
    // txtApplicationData4
    //
    this.txtApplicationData4.AutoSize = false;
    this.txtApplicationData4.Location = new System.Drawing.Point(552, 56);
    this.txtApplicationData4.Multiline = true;
    this.txtApplicationData4.Name = "txtApplicationData4";
    this.txtApplicationData4.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData4.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData4.TabIndex = 3;
    this.txtApplicationData4.Text = "";
    //
    // lblApplicationMarker4
    //
    this.lblApplicationMarker4.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblApplicationMarker4.Enabled = false;
    this.lblApplicationMarker4.Location = new System.Drawing.Point(512, 64);
    this.lblApplicationMarker4.Name = "lblApplicationMarker4";
    this.lblApplicationMarker4.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker4.TabIndex = 46;
    this.lblApplicationMarker4.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData4
    //
    this.lblApplicationData4.Location = new System.Drawing.Point(448, 64);
    this.lblApplicationData4.Name = "lblApplicationData4";
    this.lblApplicationData4.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData4.TabIndex = 45;
    this.lblApplicationData4.Text = "App Data 4:";
    //
    // txtApplicationData3
    //
    this.txtApplicationData3.AutoSize = false;
    this.txtApplicationData3.Location = new System.Drawing.Point(104, 56);
    this.txtApplicationData3.Multiline = true;
    this.txtApplicationData3.Name = "txtApplicationData3";
    this.txtApplicationData3.ScrollBars =
                System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData3.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData3.TabIndex = 2;
    this.txtApplicationData3.Text = "";
    //
    // lblApplicationMarker3
    //
    this.lblApplicationMarker3.BackColor =
                System.Drawing.SystemColors.Window;
    this.lblApplicationMarker3.Enabled = false;
    this.lblApplicationMarker3.Location = new System.Drawing.Point(64, 64);
    this.lblApplicationMarker3.Name = "lblApplicationMarker3";
    this.lblApplicationMarker3.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker3.TabIndex = 43;
    this.lblApplicationMarker3.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData3
    //
    this.lblApplicationData3.Location = new System.Drawing.Point(0, 64);
    this.lblApplicationData3.Name = "lblApplicationData3";
    this.lblApplicationData3.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData3.TabIndex = 42;
    this.lblApplicationData3.Text = "App Data 3:";
    //
    // txtApplicationData2
    //
    this.txtApplicationData2.AutoSize = false;
```

```
    this.txtApplicationData2.Location = new System.Drawing.Point(552, 11);
    this.txtApplicationData2.Multiline = true;
    this.txtApplicationData2.Name = "txtApplicationData2";
    this.txtApplicationData2.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData2.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData2.TabIndex = 1;
    this.txtApplicationData2.Text = "";
    //
    // lblApplicationMarker2
    //
    this.lblApplicationMarker2.BackColor =
            System.Drawing.SystemColors.Window;
    this.lblApplicationMarker2.Enabled = false;
    this.lblApplicationMarker2.Location = new
            System.Drawing.Point(512, 16);
    this.lblApplicationMarker2.Name = "lblApplicationMarker2";
    this.lblApplicationMarker2.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker2.TabIndex = 40;
    this.lblApplicationMarker2.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData2
    //
    this.lblApplicationData2.Location = new System.Drawing.Point(448, 16);
    this.lblApplicationData2.Name = "lblApplicationData2";
    this.lblApplicationData2.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData2.TabIndex = 39;
    this.lblApplicationData2.Text = "App Data 2:";
    //
    // txtApplicationData1
    //
    this.txtApplicationData1.AutoSize = false;
    this.txtApplicationData1.Location = new System.Drawing.Point(104, 11);
    this.txtApplicationData1.Multiline = true;
    this.txtApplicationData1.Name = "txtApplicationData1";
    this.txtApplicationData1.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
    this.txtApplicationData1.Size = new System.Drawing.Size(328, 37);
    this.txtApplicationData1.TabIndex = 0;
    this.txtApplicationData1.Text = "";
    //
    // lblApplicationMarker1
    //
    this.lblApplicationMarker1.BackColor =
            System.Drawing.SystemColors.Window;
    this.lblApplicationMarker1.Enabled = false;
    this.lblApplicationMarker1.Location = new System.Drawing.Point(64, 16);
    this.lblApplicationMarker1.Name = "lblApplicationMarker1";
    this.lblApplicationMarker1.Size = new System.Drawing.Size(32, 16);
    this.lblApplicationMarker1.TabIndex = 28;
    this.lblApplicationMarker1.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
    //
    // lblApplicationData1
    //
    this.lblApplicationData1.Location = new System.Drawing.Point(0, 16);
    this.lblApplicationData1.Name = "lblApplicationData1";
    this.lblApplicationData1.Size = new System.Drawing.Size(64, 16);
    this.lblApplicationData1.TabIndex = 27;
    this.lblApplicationData1.Text = "App Data 1:";
    //
    // tabMisc
    //
    this.tabMisc.Controls.Add(this.lblExpandMarker);
    this.tabMisc.Controls.Add(this.txtExpand);
    this.tabMisc.Controls.Add(this.lblExpand);
    this.tabMisc.Controls.Add(this.txtHierarchial);
    this.tabMisc.Controls.Add(this.lblHierarchialMarker);
```



```
    this.tabMisc.Controls.Add(this.lblHierarchial);
    this.tabMisc.Controls.Add(this.txtRestartMod8);
    this.tabMisc.Controls.Add(this.lblRestartMod8);
    this.tabMisc.Controls.Add(this.txtError);
    this.tabMisc.Controls.Add(this.lblError);
    this.tabMisc.Controls.Add(this.lblNumberLinesMarker);
    this.tabMisc.Controls.Add(this.lblRestartMarker);
    this.tabMisc.Controls.Add(this.txtNumberLines);
    this.tabMisc.Controls.Add(this.lblNumberLines);
    this.tabMisc.Controls.Add(this.txtRestart);
    this.tabMisc.Controls.Add(this.lblRestart);
    this.tabMisc.Location = new System.Drawing.Point(4, 22);
    this.tabMisc.Name = "tabMisc";
    this.tabMisc.Size = new System.Drawing.Size(888, 246);
    this.tabMisc.TabIndex = 4;
    this.tabMisc.Text = "Misc";
    //
    // lblExpandMarker
    //
    this.lblExpandMarker.BackColor = System.Drawing.SystemColors.Window;
    this.lblExpandMarker.Enabled = false;
    this.lblExpandMarker.Location = new System.Drawing.Point(112, 80);
    this.lblExpandMarker.Name = "lblExpandMarker";
    this.lblExpandMarker.Size = new System.Drawing.Size(32, 16);
    this.lblExpandMarker.TabIndex = 34;
    this.lblExpandMarker.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
    //
    // txtExpand
    //
    this.txtExpand.Location = new System.Drawing.Point(152, 80);
    this.txtExpand.Name = "txtExpand";
    this.txtExpand.Size = new System.Drawing.Size(208, 20);
    this.txtExpand.TabIndex = 32;
    this.txtExpand.Text = "";
    //
    // lblExpand
    //
    this.lblExpand.Location = new System.Drawing.Point(16, 80);
    this.lblExpand.Name = "lblExpand";
    this.lblExpand.Size = new System.Drawing.Size(96, 16);
    this.lblExpand.TabIndex = 33;
    this.lblExpand.Text = "Expand Image";
    //
    // txtHierarchial
    //
    this.txtHierarchial.AutoSize = false;
    this.txtHierarchial.Location = new System.Drawing.Point(416, 64);
    this.txtHierarchial.Multiline = true;
    this.txtHierarchial.Name = "txtHierarchial";
    this.txtHierarchial.ScrollBars =
            System.Windows.Forms.ScrollBars.Horizontal;
    this.txtHierarchial.Size = new System.Drawing.Size(464, 56);
    this.txtHierarchial.TabIndex = 29;
    this.txtHierarchial.Text = "";
    //
    // lblHierarchialMarker
    //
    this.lblHierarchialMarker.BackColor =
            System.Drawing.SystemColors.Window;
    this.lblHierarchialMarker.Enabled = false;
    this.lblHierarchialMarker.Location = new System.Drawing.Point(552, 40);
    this.lblHierarchialMarker.Name = "lblHierarchialMarker";
    this.lblHierarchialMarker.Size = new System.Drawing.Size(32, 16);
    this.lblHierarchialMarker.TabIndex = 31;
    this.lblHierarchialMarker.TextAlign =
            System.Drawing.ContentAlignment.TopCenter;
    //
    // lblHierarchial
```

```
        //
        this.lblHierarchial.Location = new System.Drawing.Point(416, 40);
        this.lblHierarchial.Name = "lblHierarchial";
        this.lblHierarchial.Size = new System.Drawing.Size(128, 16);
        this.lblHierarchial.TabIndex = 30;
        this.lblHierarchial.Text = "Hierarchial Progression:";
        //
        // txtRestartMod8
        //
        this.txtRestartMod8.Location = new System.Drawing.Point(624, 16);
        this.txtRestartMod8.Name = "txtRestartMod8";
        this.txtRestartMod8.Size = new System.Drawing.Size(72, 20);
        this.txtRestartMod8.TabIndex = 8;
        this.txtRestartMod8.Text = "";
        //
        // lblRestartMod8
        //
        this.lblRestartMod8.Location = new System.Drawing.Point(416, 16);
        this.lblRestartMod8.Name = "lblRestartMod8";
        this.lblRestartMod8.Size = new System.Drawing.Size(208, 16);
        this.lblRestartMod8.TabIndex = 7;
        this.lblRestartMod8.Text = "Restart Modulo 8 occured at byte index:";
        //
        // txtError
        //
        this.txtError.Location = new System.Drawing.Point(8, 128);
        this.txtError.Multiline = true;
        this.txtError.Name = "txtError";
        this.txtError.ScrollBars = System.Windows.Forms.ScrollBars.Horizontal;
        this.txtError.Size = new System.Drawing.Size(872, 112);
        this.txtError.TabIndex = 2;
        this.txtError.Text = "";
        //
        // lblError
        //
        this.lblError.Location = new System.Drawing.Point(16, 104);
        this.lblError.Name = "lblError";
        this.lblError.Size = new System.Drawing.Size(96, 16);
        this.lblError.TabIndex = 6;
        this.lblError.Text = "Program Errors:";
        //
        // lblNumberLinesMarker
        //
        this.lblNumberLinesMarker.BackColor =
                System.Drawing.SystemColors.Window;
        this.lblNumberLinesMarker.Enabled = false;
        this.lblNumberLinesMarker.Location = new System.Drawing.Point(112, 48);
        this.lblNumberLinesMarker.Name = "lblNumberLinesMarker";
        this.lblNumberLinesMarker.Size = new System.Drawing.Size(32, 16);
        this.lblNumberLinesMarker.TabIndex = 5;
        this.lblNumberLinesMarker.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
        //
        // lblRestartMarker
        //
        this.lblRestartMarker.BackColor = System.Drawing.SystemColors.Window;
        this.lblRestartMarker.Enabled = false;
        this.lblRestartMarker.Location = new System.Drawing.Point(112, 16);
        this.lblRestartMarker.Name = "lblRestartMarker";
        this.lblRestartMarker.Size = new System.Drawing.Size(32, 16);
        this.lblRestartMarker.TabIndex = 4;
        this.lblRestartMarker.TextAlign =
                System.Drawing.ContentAlignment.TopCenter;
        //
        // txtNumberLines
        //
        this.txtNumberLines.Location = new System.Drawing.Point(152, 48);
        this.txtNumberLines.Name = "txtNumberLines";
        this.txtNumberLines.Size = new System.Drawing.Size(208, 20);
```



```
        this.txtNumberLines.TabIndex = 1;
        this.txtNumberLines.Text = "";
        //
        // lblNumberLines
        //
        this.lblNumberLines.Location = new System.Drawing.Point(16, 48);
        this.lblNumberLines.Name = "lblNumberLines";
        this.lblNumberLines.Size = new System.Drawing.Size(96, 16);
        this.lblNumberLines.TabIndex = 2;
        this.lblNumberLines.Text = "Number of Lines:";
        //
        // txtRestart
        //
        this.txtRestart.Location = new System.Drawing.Point(152, 16);
        this.txtRestart.Name = "txtRestart";
        this.txtRestart.Size = new System.Drawing.Size(208, 20);
        this.txtRestart.TabIndex = 0;
        this.txtRestart.Text = "";
        //
        // lblRestart
        //
        this.lblRestart.Location = new System.Drawing.Point(16, 16);
        this.lblRestart.Name = "lblRestart";
        this.lblRestart.Size = new System.Drawing.Size(96, 16);
        this.lblRestart.TabIndex = 0;
        this.lblRestart.Text = "Restart Interval:";
        //
        // picManipulatedSmall
        //
        this.picManipulatedSmall.BackColor =
                System.Drawing.SystemColors.Window;
        this.picManipulatedSmall.Location = new System.Drawing.Point(456, 8);
        this.picManipulatedSmall.Name = "picManipulatedSmall";
        this.picManipulatedSmall.Size = new System.Drawing.Size(432, 344);
        this.picManipulatedSmall.TabIndex = 1;
        this.picManipulatedSmall.TabStop = false;
        this.toolTips.SetToolTip(this.picManipulatedSmall,
              "Manipulated Picture");
        //
        // picOriginalSmall
        //
        this.picOriginalSmall.BackColor = System.Drawing.SystemColors.Window;
        this.picOriginalSmall.Location = new System.Drawing.Point(8, 8);
        this.picOriginalSmall.Name = "picOriginalSmall";
        this.picOriginalSmall.Size = new System.Drawing.Size(432, 344);
        this.picOriginalSmall.TabIndex = 0;
        this.picOriginalSmall.TabStop = false;
        this.toolTips.SetToolTip(this.picOriginalSmall, "Orignal Picture");
        //
        // tabOriginal
        //
        this.tabOriginal.Controls.Add(this.picOriginal);
        this.tabOriginal.Location = new System.Drawing.Point(4, 22);
        this.tabOriginal.Name = "tabOriginal";
        this.tabOriginal.Size = new System.Drawing.Size(896, 627);
        this.tabOriginal.TabIndex = 1;
        this.tabOriginal.Text = "Original Picture";
        //
        // picOriginal
        //
        this.picOriginal.BackColor = System.Drawing.SystemColors.Window;
        this.picOriginal.Dock = System.Windows.Forms.DockStyle.Fill;
        this.picOriginal.Location = new System.Drawing.Point(0, 0);
        this.picOriginal.Name = "picOriginal";
        this.picOriginal.Size = new System.Drawing.Size(896, 627);
        this.picOriginal.TabIndex = 0;
        this.picOriginal.TabStop = false;
        //
        // tabManipulated
```

```
      //
      this.tabManipulated.Controls.Add(this.picManipulated);
      this.tabManipulated.Location = new System.Drawing.Point(4, 22);
      this.tabManipulated.Name = "tabManipulated";
      this.tabManipulated.Size = new System.Drawing.Size(896, 627);
      this.tabManipulated.TabIndex = 2;
      this.tabManipulated.Text = "Manipulated Picture";
      //
      // picManipulated
      //
      this.picManipulated.BackColor = System.Drawing.SystemColors.Window;
      this.picManipulated.Dock = System.Windows.Forms.DockStyle.Fill;
      this.picManipulated.Location = new System.Drawing.Point(0, 0);
      this.picManipulated.Name = "picManipulated";
      this.picManipulated.Size = new System.Drawing.Size(896, 627);
      this.picManipulated.TabIndex = 1;
      this.picManipulated.TabStop = false;
      //
      // openFileDialog
      //
      this.openFileDialog.Filter =
            "All files (*.*)|*.*|JPEG files (*.jpeg)" +
            "|*.jpeg|JPEG files (*.jpg)|*.jpg";
      this.openFileDialog.FilterIndex = 3;
      this.openFileDialog.Title = "Open JPEG File";
      //
      // saveFileDialog1
      //
      this.saveFileDialog1.Filter =
            "All files (*.*)|*.*|Project files (*.SEP)|*.SEP";
      this.saveFileDialog1.FilterIndex = 2;
      this.saveFileDialog1.Title = "Save SEP File";
      //
      // openFileDialog1
      //
      this.openFileDialog1.Filter =
            "All files (*.*)|*.*|Project files (*.SEP)|*.SEP";
      this.openFileDialog1.FilterIndex = 2;
      this.openFileDialog1.Title = "Open SEP File";
      //
      // timerSplash
      //
      this.timerSplash.Enabled = true;
      this.timerSplash.Tick += new
                  System.EventHandler(this.timerSplash_Tick);
      //
      // frmMain
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
      this.ClientSize = new System.Drawing.Size(904, 653);
      this.Controls.Add(this.tabMain);
      this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
      this.Menu = this.menuFrmMain;
      this.Name = "frmMain";
      this.StartPosition =
                  System.Windows.Forms.FormStartPosition.CenterScreen;
      this.Text = "ISE JPEG Manipulator";
      this.Load += new System.EventHandler(this.frmMain_Load);
      this.tabMain.ResumeLayout(false);
      this.tabConsol.ResumeLayout(false);
      this.tabSubConsole.ResumeLayout(false);
      this.tabProject.ResumeLayout(false);
      this.tabFile.ResumeLayout(false);
      this.tabHeaders.ResumeLayout(false);
      this.tabHuffman1.ResumeLayout(false);
      this.tabHuffman2.ResumeLayout(false);
      this.tabQuantizer.ResumeLayout(false);
      this.tabEncodedData.ResumeLayout(false);
      this.tabApplicationData.ResumeLayout(false);
```



```
      this.tabMisc.ResumeLayout(false);
      this.tabOriginal.ResumeLayout(false);
      this.tabManipulated.ResumeLayout(false);
      this.ResumeLayout(false);

    }

    #endregion

    /// <summary>
    /// Pre-conditions:   None.
    /// Post-conditions:
    ///    The Windows Form has been invoked.
    ///  Parameters:      None.
    ///  Return values:
    ///    Function returns void.
    ///  Description:
    ///    This function is the main entry point for a Windows based .NET
    ///    application.  This function calls the Application.Run
    ///    (System.Windows.Form) method to invoke the main form of the
    ///    application.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.Run(new frmMain());
    }


    #endregion Standard Windows From Application Methods



  }

}
```

```
///-----------------------------------------------------------------------------
///
///  File Name:         frmSplash.cs
///
///  File Description: This file implements the splash screen for the
///                    JPEG Manipulator application.
///
///  Project Name:     Selective Encryption for JPEG Images
///                    CSCI 4308-4318: Senior Project
///                    August 2003 to May 2004
///                    Department of Computer Science
///                    University of Colorado at Boulder
///
///  Project Sponsor:  Tom Lookabaugh
///                    Assistant Professor of Computer Science
///                    University of Colorado at Boulder
///
///  Project Manager:  Bruce Sanders
///                    Assistant Professor of Computer Science
///                    University of Colorado at Boulder
///
///  Team ISE Members: Shinya Daigaku
///                    Geoffrey Griffith
///                    Joe Jarchow
///                    Joseph Kadhim
///                    Andrew Pouzeshi
///
///-----------------------------------------------------------------------------
///
///  This code is open source and may be used with no cost.
///  The authors are in no way responsible for any effects
///  from the usage of this code.  It is provided as is with
///  no warranties, protections, promises or any form of
///  support.  The authors would hope it would only be used
///  for good purposes.  Thank you.
///
///-----------------------------------------------------------------------------
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace JPEG_Manipulator
{
  /// <summary>
  /// Summary description for frmSplash.
  /// </summary>
  public class frmSplash : System.Windows.Forms.Form
  {
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;

    public frmSplash()
    {
      InitializeComponent();
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
        if(components != null)
        {
```



```
          components.Dispose();
        }
      }
      base.Dispose( disposing );
    }

    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      System.Resources.ResourceManager resources = new
        System.Resources.ResourceManager(typeof(frmSplash));
      //
      // frmSplash
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
      this.BackgroundImage = ((System.Drawing.Image)
        (resources.GetObject("$this.BackgroundImage")));
      this.ClientSize = new System.Drawing.Size(512, 280);
      this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
      this.Icon = ((System.Drawing.Icon)
        (resources.GetObject("$this.Icon")));
      this.Name = "frmSplash";
      this.StartPosition =
        System.Windows.Forms.FormStartPosition.CenterScreen;
      this.Text = "frmSplash";
      this.TopMost = true;

    }
    #endregion
  }
}
```