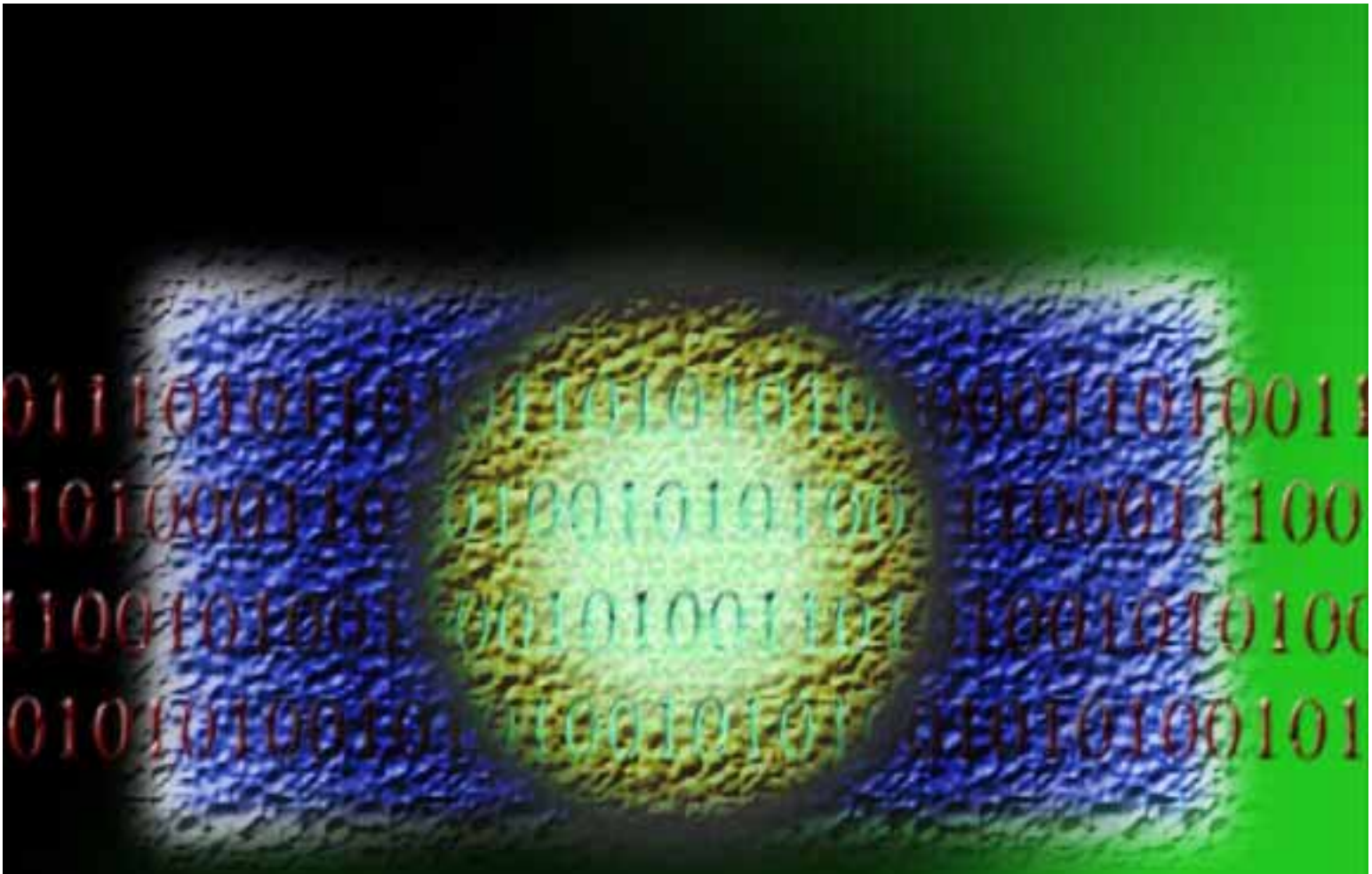


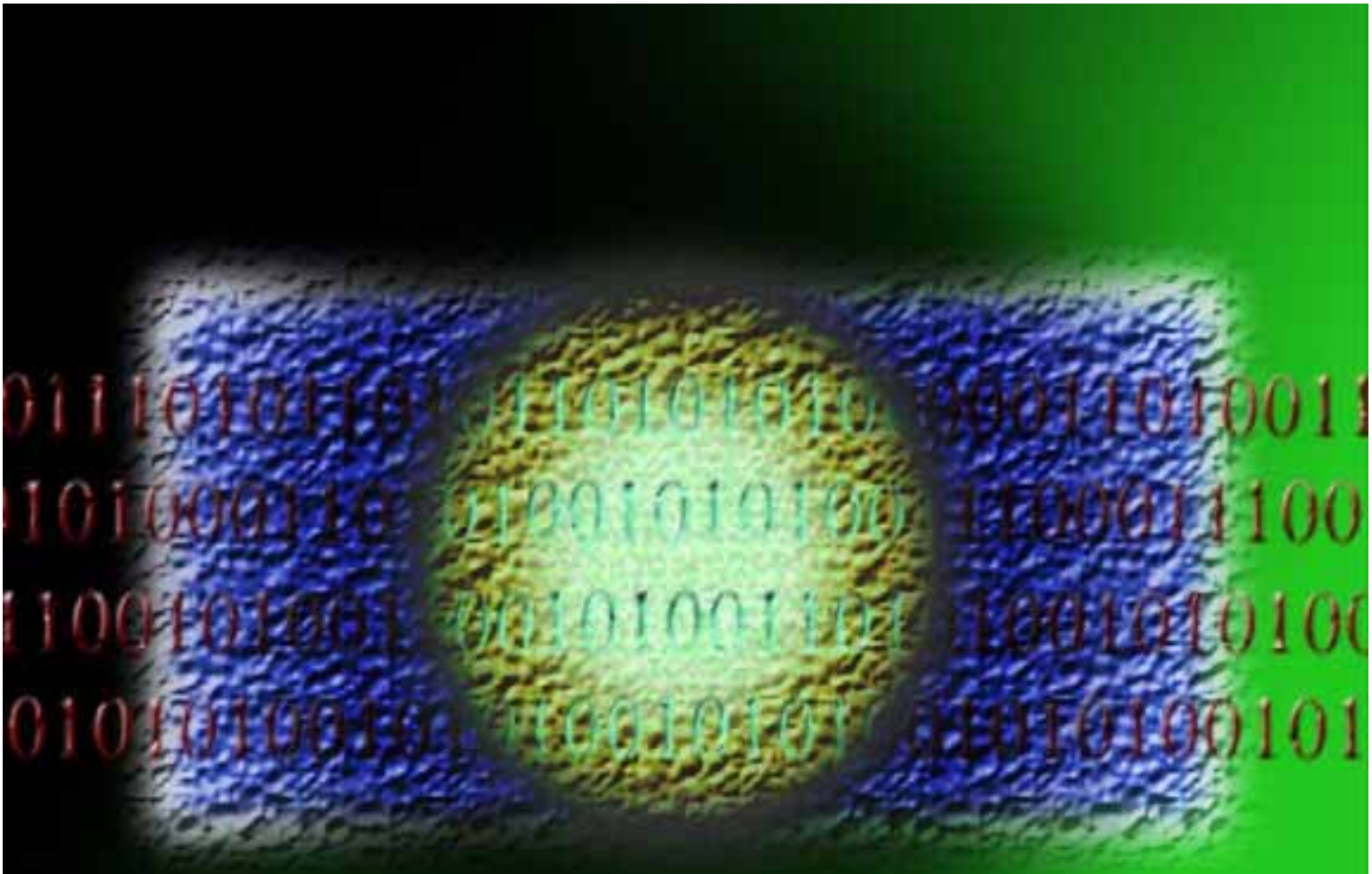
Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309-0430

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science
Department of Computer Science
University of Colorado at Boulder
430 UCB
Boulder, CO 80309-0430



Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309-0430

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science
Department of Computer Science
University of Colorado at Boulder
430 UCB
Boulder, CO 80309-0430

Signatures

Tom Lookabaugh
University of Colorado at Boulder
Department of Computer Science
430 UCB
Boulder, CO 80309

Bruce Sanders
University of Colorado at Boulder
Department of Computer Science
430 UCB
Boulder, CO 80309

Acknowledgements

We would like to thank a number people for their contributions and aid throughout the course of this project. The first is Dr. Tom Lookabaugh, who sponsored our project. Dr. Lookabaugh took the time to teach us about selective encryption and the JPEG compression. He has guided this project through every phase and we hope our final product satisfies his expectations. We would also like to thank Dr. Bruce Sanders for keeping the project on track, his invaluable input, and for believing in the success of Team ISE. Third, we thank Martin Cochran our TA for the Senior Project. He was required to read every one of our documents (multiple times) and provided critical insight into each. We thank David Keaton, who aided us in the lab and with the insides of the Linux systems. Then we would like to thank Cecilia M. Girz who aided us in proofreading our research paper. Lastly, we could not have made it this far without Jamie Griffith. Thank you for your wonderful support.

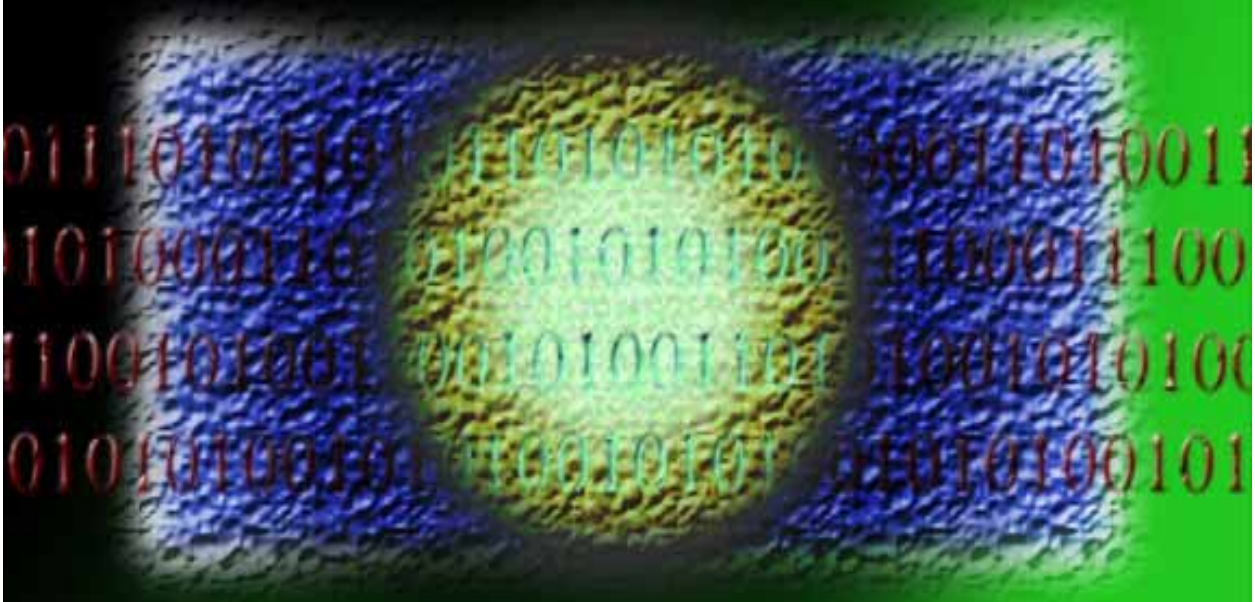
Contents

Project Proposal

Team ISE

Image Selective Encryption

Project Proposal
September 2003



Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Project Proposal

A constant amount of traffic flows between computers connected to the Internet. A large volume of information may take a long time traveling from destination to destination. The resulting speed reduction makes it desirable to compress the file as much as possible in order to send the smallest amount of data. Compression of data has allowed for the high-speed data transfers that have made Internet communication and business very workable.

In addition to sending the smallest amount of information possible, users also attempt to maintain a certain level of security upon their information. Due to the fact that common encryption methods generally manipulate an entire file, most encryption algorithms tend to make the transfer of information more costly in terms of time and bandwidth. Thus, users pay a price for security relative to their desired level of security. One possible solution would be a system of encryption that works cooperatively with the standard compression schemes. *Selective Encryption* of only a small percentage of the file's bits will facilitate this solution. Because most encryption schemes will make the file larger, selective encryption seeks only to encrypt portions of the file that will make it unusable. In other words, if a user does not have the proper decryption device, the file should not be usable. Selective encryption will minimize the necessary increase in file size due to encryption while maintaining a maximum level of uselessness, or damage, to the product.

An image could be encrypted with any of the sufficiently secure encryption algorithms available to the open source community, but this will usually result in a dramatic increase in file size that will severely increase transfer time over the Internet. However, selecting key parts of a file for encryption and only encoding those bits can actually render an image unusable. The initial statistical analysis done by the team will consist of specifically breaking down the standard JPEG compression scheme into its usable parts and evaluate which of these, if encrypted, will cause a potential user to pay for rights to the image or force subscription to the provider service.

Team ISE (Image Selective Encryption) will deliver a package for selectively encrypting JPEG (Joint Photographic Experts Group) still image files. The package will provide the tools necessary to encrypt the critical information of a JPEG file in cooperation with existing standard compression tools. This package will handle JPEG files in such a way that only a small percentage of the total file will be encrypted. Selective Encryption security will not extend to the level of military secrecy, but rather a level that would deter all but brute force attacks, allowing users to securely protect private JPEG images.

An additional aspect of the encryption analysis will be the determination of the specific targets in the file for encryption. For example in an MPEG file there are headers that contain a small portion of the overall number of bits but which are extremely vital to the reproduction of the movie by the user. So, if certain headers were to be encrypted the percentage of the file being manipulated would be less than ten percent of the total number of bits in the file. Although only a small portion will be encrypted, the resulting

damage experienced by an unauthorized user would be sufficient to cause the user to pay for the decryption package. However, there are other targets that, while they can be encrypted and will do sufficient damage, can be guessed by an attacker. The attacker could, with some degree off effort, render the file useful without use of the decryption software. For example, if the frame rate of an MPEG file was encrypted, an attacker could try all three of most common frame rates and one of these is certain to produce the correct rate for the particular video. In the case of JPEG Selective Encryption, Team ISE will have to balance the targets for encryption against ease of simple attacks.

A permanent website will be constructed by the team to make the software package available to anyone interested in the software process. As it is vital to the world of cryptography to let the community view the approach, the first form of the working prototype will be made available on the website. From this, feedback can be received not only from the team itself, but also from the cryptography community at large.

So, following the guidelines of the ongoing MPEG research (also being guided by the sponsor), the team will study the JPEG process and earlier attempts at encryption. With the sponsor's assistance, Team ISE will devise a workable approach to handling individual JPEG images following the concept of selective encryption.

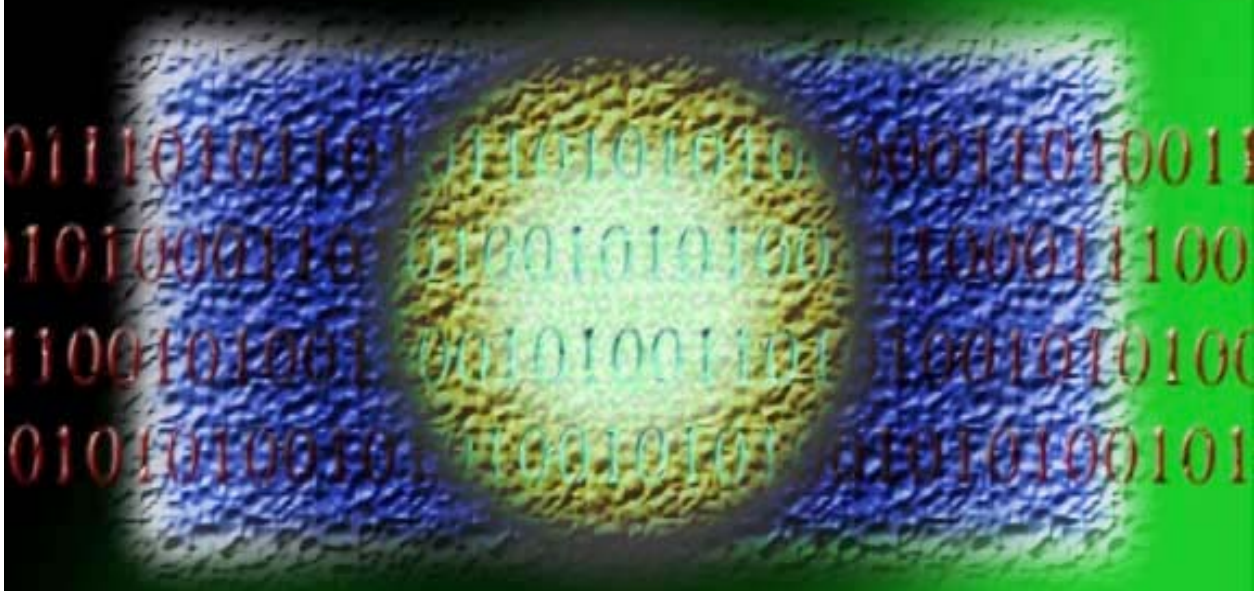
It is possible that the team will complete the JPEG process early enough in the year that they will able to apply the same approach to other types of compressed files (text, audio, etc.) However, this initial specifications document applies only to the envisioned JPEG project.

Initial Requirements

Team ISE

Image Selective Encryption

Initial Requirements
September 19th, 2003



Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffery Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Associate Professor of Computer Science

Project Proposal

A constant amount of traffic flows between computers connected to the Internet. A large volume of information may take a long time traveling from destination to destination. The resulting speed reduction makes it desirable to compress the file as much as possible in order to send the smallest amount of data. Compression of data has allowed for the high-speed data transfers that have made Internet communication and business very workable.

In addition to sending the smallest amount of information possible, users also attempt to maintain a certain level of security upon their information. Due to the fact that common encryption methods generally manipulate an entire file, most encryption algorithms tend to make the transfer of information more costly in terms of time and bandwidth. Thus, users pay a price for security relative to their desired level of security. One possible solution would be a system of encryption that works cooperatively with the standard compression schemes – *Selective Encryption* of only a small percentage of the file's bits. Because most encryption schemes will make the file larger, selective encryption seeks only to encrypt portions of the file that will make it unusable. In other words, if a user does not have the proper decryption device, the file should not be usable. Selective encryption will seek to balance the necessary increase in file size, or bandwidth, due to encryption while maintaining a maximum level of uselessness, or damage, to the product.

An image could be encrypted with any of the sufficiently secure encryption algorithms available to the open source community, but this will usually result in a dramatic increase in file size that will prohibit transfer over the Internet. However, selecting key parts of a file for encryption and only encoding those bits can actually render an image unusable. The initial statistical analysis done by the team will consist of specifically breaking down the standard JPEG compression scheme into its usable parts, and evaluating which of these, if encrypted, will cause a potential user to pay for or subscribe to the decryption service.

Team ISE (Image Selective Encryption) will deliver a package for selectively encrypting JPEG still image files. The package will provide the tools necessary to encrypt the critical information of a JPEG file in cooperation with existing standard compression tools. This package will handle JPEG files in such a way that only a small percentage of the total file will be encrypted. The level of encryption will not reach to the height of military secrecy, but rather a level that would thwart most simple attacks while causing potential users to pay for viewing the image.

An additional aspect of the encryption analysis will be the determination of the specific targets in the file for encryption. For example in an MPEG file there are headers that contain a small portion of the overall number of bits but which are extremely vital to the reproduction of the movie by the user. So, if certain headers were to be encrypted the

percentage of the file being manipulated would be less than 10% of the total number of bits in the file. Although only a small portion will be encrypted, the resulting damage experienced by an unauthorized user would be sufficient to cause the user to pay for the decryption package. However, there are other targets that, while they can be encrypted and will do sufficient damage, can be guessed at by an attacker. The attacker could, with some degree off effort, render the file useful without use of the decryption software. For example if one encrypted the frame rate of an MPEG file, an attacker could just guess at the 3 most common frame rates, and one is certain to produce a correct copy of the video. Again, Team ISE will have to balance the targets for encryption against ease of simple attacks.

A permanent website will be constructed by the team to make the software package available to anyone interested in the software process. As it is vital to the world of cryptography to let the community view the approach, the first form of the working prototype will be made available on the website. From this, feedback can be received not only from the team itself, but also from the cryptography community at large.

So, following the guidelines of the ongoing MPEG research (also being guided by the Sponsor), the team will study the JPEG process and earlier attempts at encryption. With the Sponsor's assistance, Team ISE will devise a workable approach to handling individual JPEG images following the concept of selective encryption.

It is possible that the team will complete the JPEG process early enough in the year that they will able to apply the same approach to other types of compressed files (text, audio, etc.) However, this initial specifications document applies only to the envisioned JPEG project.

Table of Contents

1. INTRODUCTION

The goal of selective encryption is to minimize the amount of encryption applied to a file while maximizing the damage done to the image being viewed by a user not in possession of the authorized decryption package. Complete encryption is not a requirement of the process, nor is rendering the file to useless to the level of complete military secrecy. It is acceptable for an attacker to be able to view portions of the file; however the file should be distorted enough that an attacker would not wish to use the encrypted file but would rather purchase or subscribe to the decryption method for access to the original files.

Multimedia files prove to be a good subject for selective encryption. This is due to the fact that the multimedia files tend to be very large and their compression algorithms concentrate critical information in small portions of this bit stream. If the critical information is encrypted, the remaining information becomes useless to those without the proper decoder. There are many types of compression algorithms that fit this description. Examples of such are MPEG 1, 2 and 4 video, AAC audio, G.723 and G.729 video, and JPEG and JPEG2000 image.

The focus of this project is to research and develop an algorithm for selective encryption of a standard baseline compressed JPEG image file. This process must encrypt a file in such a way that the amount of the file being encrypted is relatively small, yet the damage done to the file is on a scale that would render the file useless without a proper decryption device. This process will be delivered in a package that will include an encrypter for JPEG files and a decrypter that will reverse the operation. This package will be made available in a fully open source form on the website that will be constructed by the team.

The website is to be constructed on a server being purchased by the Sponsor in an environment that will match the other computers in the working lab. The team will acquire a fixed IP address from the proper University authorities and will set up a simple website capable of informing viewers about the possibilities of the technology of Selective Encryption and to provide them with a package they can download and test. The site will provide links to important information and will remain up permanently even once the project is complete.

The envisioned software package will accomplish a seemingly simple result while being extremely effective and usable to the appropriate users. Below is a flow chart showing the general picture of the package's operations. (Figure 1.1.)

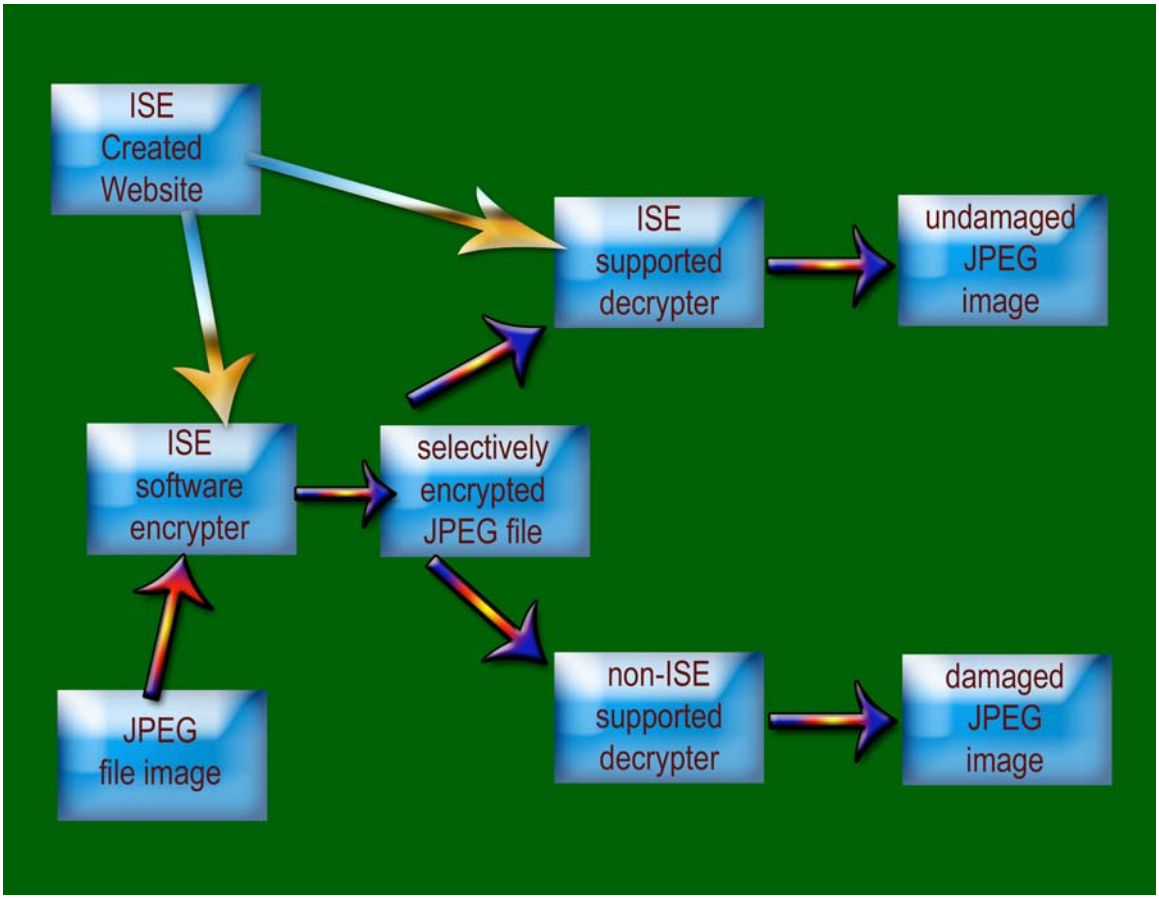


Figure 1.1: Conceptual Overview of ISE Software

The ISE website displayed in the flow chart will be used to distribute the ISE software and will also contain information on the product as well as the research behind it.

The list of requirements for ISE follows. As there is a degree of research that must be done by the team under the Sponsor's guidance and supervision, the general path of the research is given as a precursor to the actual final product requirements. Further, as this research will to some degree determine the final necessary requirements, this document will serve as a starting point for the project, but will be refined later.

2. RESEARCH PATH

2.1. Research and Analysis Requirements

The research and analysis will be the initial part of this project. The final product of this process is essentially a completely determined approach.

- Proportional Analysis of a large quantity of JPEG images to define what might be acceptable targets within the JPEG file structure for encryption.
- Analysis of earlier methods of encryption for performance and effectiveness.

- Analysis of different encryption methods and targets in the JPEG image file for percentage of file encryption vs. image corruption.
- Analysis of different encryption methods and targets in the JPEG image file for the encryption target's susceptibility to attack.
- Final stage of the research analysis will evaluate and get approved by the Sponsor an acceptable performance evaluation taking into account all necessary factors that the research will review.

3. REQUIREMENTS

The requirements have been divided into several logical sections. These sections include the requirements of the Supporting Environment, Functional Requirements, Performance Requirements, and Documentation and Release Requirements.

➢ 3.1. Supporting Environment

The supporting environment includes specification of both the expected environments that the package should be able to perform in and the form in which the package will be written. There is also a basic specification of the hardware environment the package will require to be run in.

- **3.1.1. Software**
  - Package to be operational in Linux Red Hat 9.0, Windows XP and Mac OS X.
  - Package to be written in ANSI C/C++ incorporating the Independent JPEG Group (IJG) package.
  - Package should not change IJG's claim of wide portability (see http://www.ijg.org for specific environments.)
  - Web page will be built on a server and OS supplied by the Sponsor.
    - ✓ Web page to be viewable on Internet Explorer 6 and Safari 1.0.
    - ✓ Web page will use HTML version 4.01.

- **3.1.2. Hardware**
  - Package should be able to be run on any computer system supporting color graphics.
  - Generic color monitor and JPEG image viewing system outlined above.
  - Mouse, and Keyboard.
  - Hardware supports the software environment outlined above.

➢ 3.2. Functional Requirements

Functional Requirements specify all of the functionality that ISE is required to provide. This includes functionality interfacing to the software package, and the

commands supported by the software package. The requirements of the web page created to support the package will also be listed in this section.

- **3.2.1. Required Operations**
    - Encrypt a standard image file in cooperation with the standard JPEG encoding format.
    - Maintain compliance to the JPEG compression standard.
    - Decrypt a standard compressed JPEG image file in cooperation with the Standard JPEG decoding format.
    - Level of encryption is not "secretive/military" but only to level of damage that would force subscription to image viewing.
    - Time permitting; the package will also selectively encrypt audio files, possible mp3 or AC3, and/or text files, such as zip files. However, these are secondary options. The main goal of the project is to deliver a package that selectively encrypts JPEG files.
        - Any attempt at these secondary projects would follow the same line of research into implementation.

- **3.2.2. Interface to Package**

    - Must be able to read in either .jpg or .bmp files for encryption.
    - Research will determine what file type the encryption module will output.
    - Decrypt module will input the appropriate file type.
    - Decrypt will output a standard .jpg file.
    - Final product will be a software package with command line user interface and appropriate incorporation into the standard JPEG tools.

- **3.2.3. Commands for software package**

    - Encrypt -- take a standard .jpg or .bmp file and convert to an encrypted JPEG file.
    - Decrypt -- take an encrypted JPEG file and convert to a standard .jpg file.

- **3.2.4 Supporting Web Page**

    - To be built on server and OS provided by Sponsor.
    - Contain links explaining the purpose of the software package provided by Team ISE.
    - Contain links to downloadable version of the software package.
    - Contain links to the software documentation as well as providing the user with the ability to download the documentation.
    - Contain open source files of the software package.
    - Contain links to other sources of related information.

➢ 3.3. Documentation and Release Requirements

The following requirements specify the documentation that is to be provided, along with issues related to the release and delivery of the final product.

- **3.3.1. Documentation Requirements**

  - Man Page -- standard UNIX man page.
  - User Tutorial -- presentation of system for first-time user.
  - Research paper written up in style of the Sponsor's MPEG reference paper.
  - Web site to include all code and documentation and supporting links.

- **3.3.2. Release Requirements**

  - Delivered as zipped files for Unix, Windows and Mac users.
  - File will include entire source tree of software
  - File will include installation programs for automatic generation and installation of executable and preview/evaluation programs.
  - Documentation provided only on the website for download.

4. SUMMARY

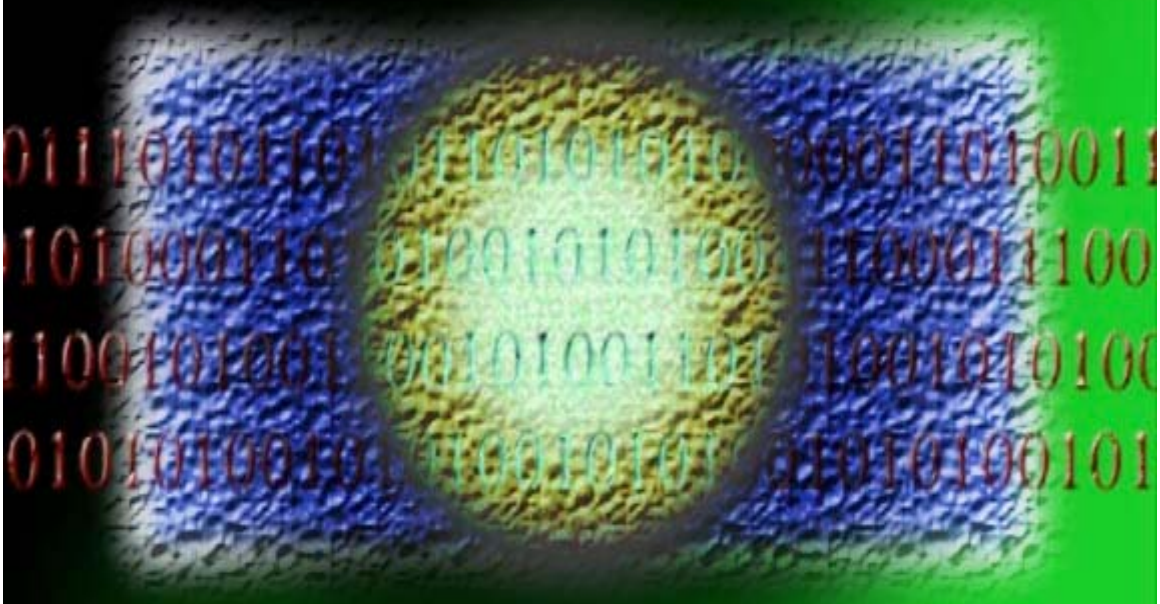
The purpose of this document was to give an initial outline for the path of research and the set of requirements for the ISE software package. These requirements include the software and hardware environments the application will run on, the functional requirements, the research and analytic requirements, and the supporting and research document's requirements that will be included along with the software package. These requirements will be modified at a later date when more information is known about completing the software package.

System Architecture

Team ISE

Image Selective Encryption

System Architecture
September 30th, 2003



Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffery Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Project Proposal

The selective encryption project (Team ISE) is being sponsored by Assistant Professor of Computer Science, Tom Lookabaugh. Dr. Lookabaugh teaches and researches in the technology and practice of video communication, high technology businesses, and the intersection of policy, innovation, and management. His website contains a great deal of information on his research projects and responsibilities: http://itd.colorado.edu/lookabaugh/.

While many compression techniques have allowed an increase in the flow of traffic across the lines of the Internet, the files they produce are largely unprotected by efficient security measures. They are generally unencrypted and susceptible to unauthorized viewing. Team ISE will be working to incorporate encryption into common compression schemes starting with the JPEG image standard. While the final product is not required to provide more than the classes that would define the encryption and decryption methods, the initial portion of the project is oriented around the research and analysis of the most workable methods for securing compressed files. For this we will be developing a preview and testing suite with a simple graphical interface providing the ability to attack different portions of the compression standard.

The immediate efforts of the team will focus on developing selective encryption for the JPEG standard. If that portion of the project is able to be finished in a reasonable period of time, the team will venture into developing schemes for audio and text compression standards (MP3, zip, etc.)

Therefore the design of the test suite will first be for JPEG development. The test suite will utilize a pattern or process that can easily be extended to other desired formats.

Finally, the team will construct a permanent website which will allow anyone to download the team's previews, products, code and documentation. The site will be constructed on a computer and operating system provided by the Sponsor.

Table of Contents

1. INTRODUCTION

Team ISE is being sponsored by Assistant Professor of Computer Science, Tom Lookabaugh: http://itd.colorado.edu/lookabaugh/.

Selective encryption is intended to utilize the standard formatting of commonly used compression schemes. Targeting small portions of a file that has been or will be divided into pieces defined by the standard algorithm can allow encryption of only a tiny portion of the file. If the target is chosen with care, the encryption can have the effect of damaging the usability of the file for the user who does not have the compatible decryption package.

Team ISE will first be developing a selective encryption scheme for the JPEG image standard. A standard encryption algorithm will be used to encrypt target portions of the file. However, because it is not the goal of Team ISE's project, the team will not be developing or implementing the encryption algorithm. However, the team will include a freely available encryption implementation with the software package. Current encryption candidates are the RC4 stream cipher algorithm and the AES block cipher. However, the team is not limited to these options.

The final product that the team will be providing to the open source community will be methods or classes that will provide the ability to encrypt and decrypt a file created by or used with a standard compression method. These methods or classes will be written in ANSI C/C++. See Figure 1.1 for an overview of the usage of the team's final product. Given a reasonable amount of time the team will also attempt to create selective encryption schemes for other compression standards, such as audio and text.

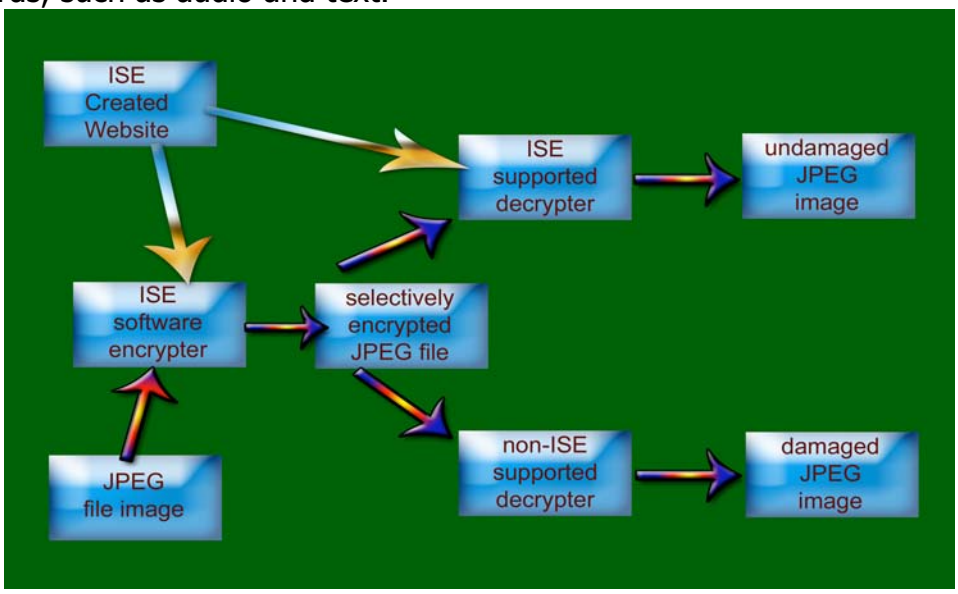


Figure 1.1: Conceptual Overview of ISE Software

To develop this and possibly other products, the team will be creating a test suite for use in establishing a workable encryption scheme. Again, there will be no work by the team to create an encryption algorithm; the target is only the development of a scheme for selective encryption. The intention of selective encryption is that it be such a system that it is possible to use any standard encryption algorithm. The test suite will effectively simulate an end user product. It will utilize a standard encryption algorithm but the end user would not be required to use any algorithm chosen by the team.

There are several necessary functions that the test suite must have. It will first be able to preview a standard file. Each compressed file is divided into separate pieces of information as per the compression standard. Therefore, the test suite will provide the ability to manipulate the various portions of the compression standard in each compressed file. Having manipulated the file, the test suite will be able to preview the encryption attempt without the benefit of compatible decryption. It will also have the ability to preview a standard file that has been both encrypted and decrypted. The decryption options will allow the user try to defeat the encryption methods (let the user put on a black hat.) Any selective encryption scheme could be developed using a package that implemented these features.

The test suite will be developed with Visual Studio C#.

The test suite will use the encryption and decryption classes or methods that the team is developing. The methods will be developed in standard ANSI C/C++, as per the specifications document, and will be able to be called by the test suite.

The website that will be constructed by the team will be on a computer and operating system provided by the Sponsor. It will have a simple home page with links to previews, final product code and to documentation.

This document will primarily define the high-level design architecture for the final product, the test suite, and for the website to be used and developed by the team. For each element of the project, this document will outline the design of invocation, user interface, high-level modular decomposition and file description.

2. INVOCATION

Throughout this document design specifications will be laid out for the final product, the test suite and the website. There will be more or less detail depending on the necessary complexity of the object being described.

2.1. Production Code:

- On invocation, the ISE encrypter will be given an image file path, a flag indicating the target portion(s) of the image for encryption, a step size value for the quality of encryption (the amount or portion of the file to be encrypted), the encryption key file path, (optional) the output file name and path.
- On invocation, the ISE decrypter will be given an encrypted image file path, a flag indicating the target portion(s) of the image for decryption, a step size value for the quality of decryption (the amount or portion fo the file to be decrypted), the decryption key file path, (optional) the output file name and path.

2.2. Test Suite:

- The Previewer will start up as a version 1.1 .NET windowed application using a default test image and an appropriate set of default parameters for the encryption and decryption modules.

2.3. Web Site:

- The web site will be accessed through a fixed IP on the University of Colorado network and will have a home page that will identify the project and provide links to previews, final product code and documentation.

3. USER INTERFACE

The general high-level design for the various user interfaces will be laid out as follows.

3.1. Production Code:

- The user interface will be strictly a command line environment where the command encryption or decryption is given along with the necessary parameters.

  Parameters:
  - int file_type_encryption (image_file_path, target_flag(s), encryption_quality, key_file_path, output_file_path)
  - int file_type_decryption (image_file_path, target_flag(s), decryption_quality, key_file_path, output_file_path)

3.2. Test Suite:

- The test suite will be constructed in Visual Studio C# (see Figure 3.1 below) and will attempt to make development of the selective encryption scheme very organized and straightforward.

- There will be a tab corresponding to each major portion of the compression standard. In the JPEG standard, the compressed file is stored in easily parsible portions and each has a clear identification and purpose (Huffman encoding tables, Quantizer tables, etc..)
- The test suite will have a modular design which will allow the team to scale it to work for other compression formats, such as MP3 and ".zip".

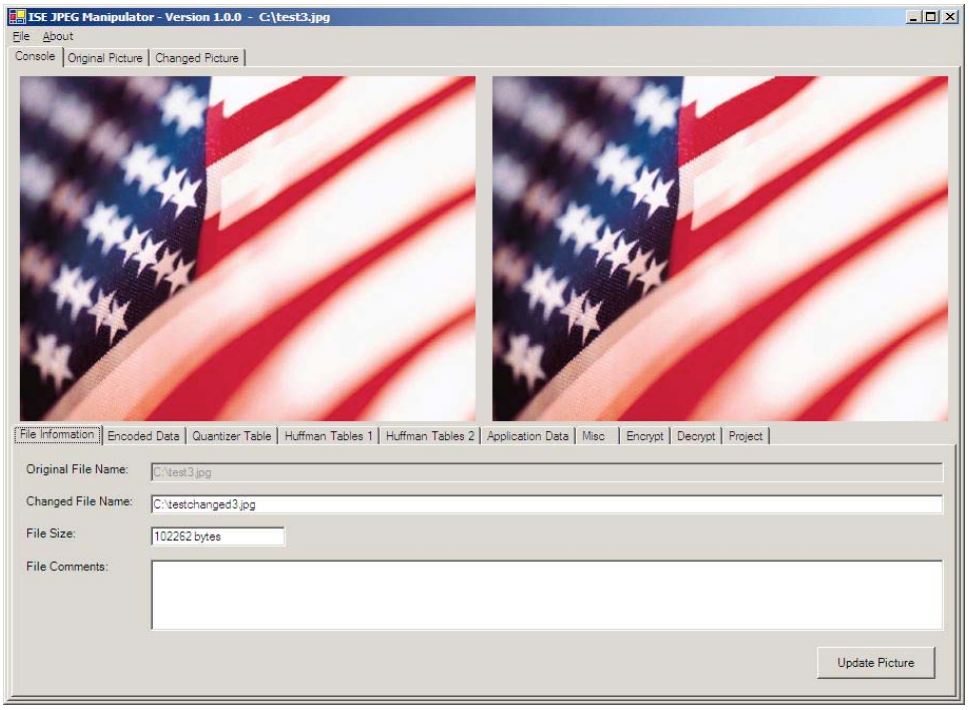


Figure 3.1: Screen Shot of ISE Testing Suite being prototyped

Graphical User interface:

(Comments will only be placed under the items not obvious.)

- Menu Bar
  - File
    - Open project
    - Save project
    - Save project as
    - Exit
  - Help
    - Help
    - About (Will include versioning information.)
- Image Display section
  - Console tab
    - Original image preview
      - Displays the original unmanipulated image.
    - Final output image preview

- Displays the final image after encrypt and decrypt.

- File information tab
  - Will include the original file name and a button to allow opening another file.
  - Will display the output file name and will include a browse button so that the output name can be changed without overwriting any existing files.
  - Will display file size.
  - Will include a section for adding comments to the output file.
- Huffman Encoded Scan Data tab
  - Will display the Scan header.
  - Will display the encoded data (start of scan).
  - If the file is manipulated, this tab will display the original header and the original encoded data for comparison.
- Quantizer Table tab
  - Will display up to 5 quantizer tables and if any are modified will also display the unmanipulated table.
- Huffman Table tab
  - Will display up to 5 Huffman tables and, if any are modified, will also display the unmanipulated tables.
- Application Data tab
  - Fields for up to 10 of the available application data flags.
- Miscellaneous data tab

  Fields for:
  - Restart Interval
  - Number of Lines
  - Expand Image
  - Restart modulo 8 occurred at byte index
  - Hierarchicial Progression
  - Program Errors
- Encryption tab
  - Will have check boxes for all possible flags.
  - Will have radio buttons for any implemented encryption methods.
  - Will display the path to the encryption key and will allow this path to be set or browsed.

        - Will have a field to define the step size for the quality of encryption.
        - Will call the encrypt function outlined in section 3.1.
    - Decryption tab
        - Will have check boxes for all possible flags.
        - Will have radio buttons for any implemented decryption methods.
        - Will display the path to the decryption key and will allow this path to be set or browsed.
        - Will have a field to define the step size for the quality of the decryption.
        - Will call the decrypt function outlined in section 3.1.
    - Project comments tab
        - Will have a field for project comments to be entered and saved with the project.
- Original picture tab
    - Will display the original picture without the size alterations made in the display window.
- Final Image tab
    - Will display the encrypted image without the size alterations made in the display window.

3.3. Web Site:

- The web site will be a very simple construction with a home page directing users to previews, final product code, documentation and test suite.

Figure 3.3.1: Screenshot of ISE Web Page

4. HIGH-LEVEL MODULAR DECOMPOSITION

A high-level modular decomposition of Team ISE's software project is presented in Figure 1.1. The project consists of four main modules:

- ISE Website
- ISE Encryptor
- ISE Decryptor
- Test Suite

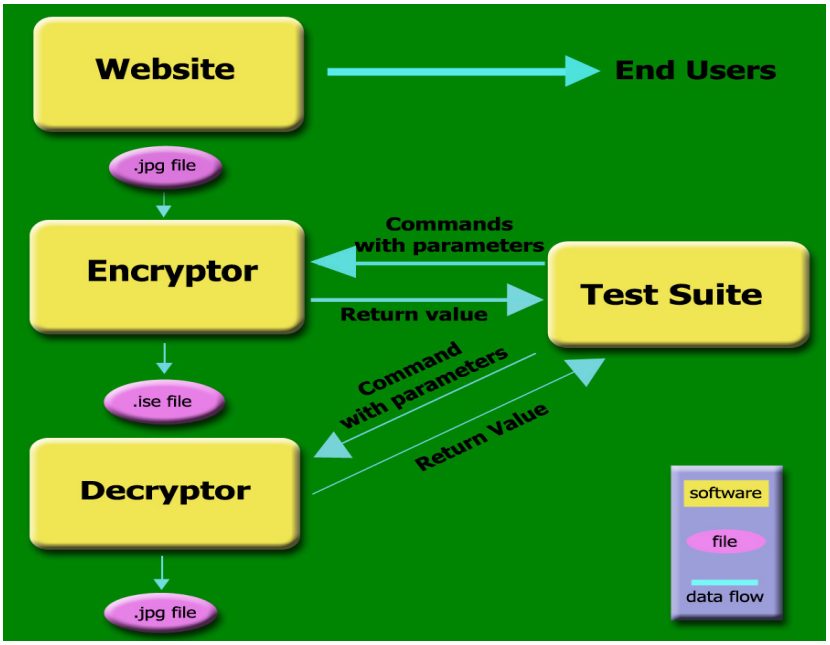


Figure 4.1 High level modular decomposition of ISE

Any comments in Sections 4.2 through 4.4 that seem to apply only to JPEG images can and will be adopted to any other compression standards that the team may attempt during the project.

4.1 ISE Website

- The website will serve as the distributor for Team ISE's software package.
- It will also include links to all documentation provided by the team about the software package and the research behind the implementation.
- The website will display the product and output. This will either be done with a screen shot or a possible test encryption service found at the site.

4.2 ISE Encryptor

- The ISE encryptor will be invoked by a command line call.
- The encryptor will take a few parameters, namely the JPEG filepath, a flag indicating the target portion of the file for encryption, a step size value to define the quality of encryption, the pathname of the location of the key used in the encryption, and an optional output file name and path.
- The encryptor will gracefully terminate if the file imported to it does not end in a ".jpg" extension.
- If the optional file name and path is not included, the encryptor will produce an encrypted file in the current directory with the same name as the orignal JPEG file, however it will contain a ".ise" extension.
- The encryptor module will allow the user to determine which portion(s) of the JPEG file they would like to be encrypted. The portions of the file that can be targeted are defined by the compression standard (these are outlined in Section 3.2 for the JPEG standard.)
- The Module will allow adjustment of the desired quality of encryption. This will vary in implementation between standards. For some formats this will be a percentage of the file to be encrypted. For other formats this might define what portions of the file are to be encrypted.

4.3 ISE Decryptor

- The ISE decryptor will also be invoked by a command line call.
- The decryptor will take the following parameters: encrypted image file path, a flag indicating the target portion of the image for decryption, a step size to define the quality of decryption, the decryption key file path, (optional) the output file name and path.
- Like the encryptor, if an output file name and path is not specified, the decryptor will produce a standard JPEG file with the same name as the encrypted file, however the ".ise" extension will revert back to a ".jpg" extension and a number will be assigned to the end of the file name string("dog.ise" will become "dog001.jpg".)
- The decryptor will gracefully terminate if it is run on a file without the ".ise" extension.
- The decryptor module will allow the user to determine which portion(s) of the file they would like to be decrypted. The portions of the file that can be targetted are defined by the compression standard (these are outlined in Section 3.2 for the JPEG standard.)
- The Module will allow adjustment of the desired quality of decryption. In most cases this would be required to match the encryption step size setting.

4.4 ISE Test Suite

- The ISE Test Suite will provide the team with valuable information about the contents of the compressed JPEG file before and after encryption.
- The Test Suite will also be available to users who wish to view the file changes that can be made to JPEG files using selective encryption.
- It will also display the original and final JPEG images side by side allowing the user to visually compare the differences in image quality. The test suite will implement and include all of the functionality described in section 3.2 for the JPEG standard.

5. FILE DESCRIPTIONS

There are several files that will be used by Team ISE's software package. They will be divided into the following categories:

- Input Files
- Output Files
- Test Suite Files
- Optional Project Extension Files

Again, any comments in the following sections that seem to apply only to the JPEG image standard will be adopted by the team and applied to any other compression standards attempted by the team during the project.

5.1 Input Files

- The encryptor will require standard JPEG files. The file will have to end in ".jpg" and will have to be a standardly recognizable JPEG image.
- The decryptor will require files that have been output by the ISE encryptor ending in the ".ise" extension.

5.2 Output Files

- The encryptor will produce encrypted JPEG files ending in a ".ise" extension
- The decryptor will produce standard JPEG files ending in a ".jpg" extension

5.3 Test Suite Files

- The test suite will require standard JPEG files. The file will have to end in ".jpg" and will have to be a standardly recognizable JPEG image.

5.4 Optional Project Extension Files

- Time permitting, Team ISE will provide encryption and decryption modules to selectively encrypt other file formats, for example MP3 or ".zip" files. In this case, the encryptor will work on files ending in the standard extensions for these compression methods, and will produce selectively encrypted files with ."ise" extensions. The decryptor module will work on files with this new extension and reproduce the original files with their standard file extensions. Again, this is potential additional work to be performed by Team ISE. The main goal of the project is the production of JPEG encryption and decryption modules.

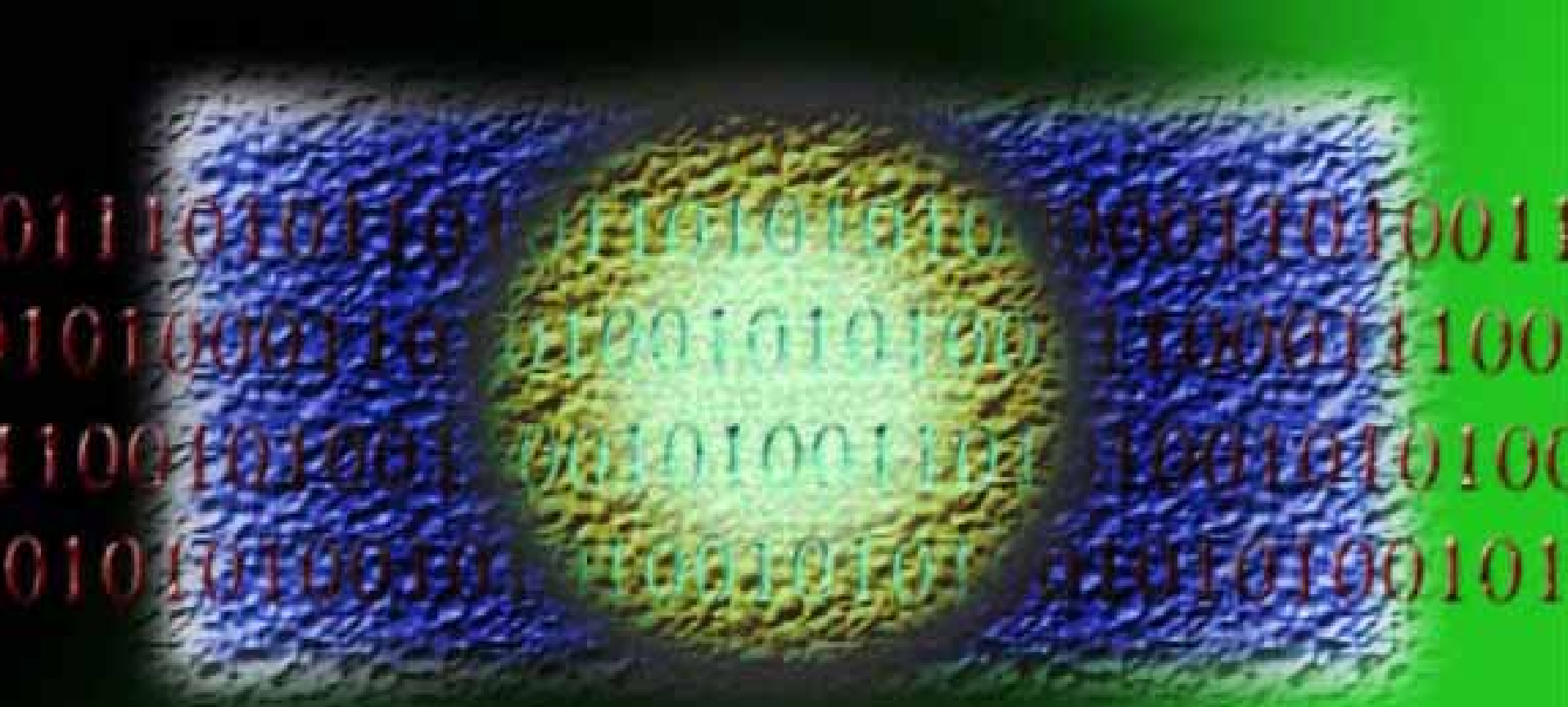
6. SUMMARY

This has been a very high-level view of the initial thoughts on a system architecture for Team ISE's selective encryption project. The document includes information on the architecture of the final encryption and decryption modules, the team's distribution website, as well as extensive planning on important tools which will be implemented for the team's reseach, development and testing. These tools will be available to users who wish to view the inner workings of the selective encryption methods. Thought was also given to the scalability of the project, specifically the inclusion of encryption and decryption modules for compression standards other than JPEG. The design and architecture allows for the extension of the available modules to other formats. It should serve as a strong beginning from which the team can start prototyping. It will also allow the team to begin the formulation of a more detailed design of the product.

Overview Presentation

Team ISE

Image Selective Encryption



Team ISE

Image Selective Encryption

Team ISE

Image Selective Encryption

Joe Jarchow

Shinya Daigaku

Joseph Kadhim

Andrew Pouzeshi

Geoffrey Griffith

Sponsor

- **Tom Lookabaugh**
  - **Assistant Professor in Computer Science Department**
  - **Faculty Director of Interdisciplinary Telecommunications**
  - **Research into areas such as**
    - **Selective Encryption**
    - **Broadband**
    - **Multimedia and Distance Learning**

Joe J

Problem:

- **Multimedia files are often very large**
- **Encrypting can require extensive processing time**
- **Can also increase the file size**
- **No current intelligent method for securing multimedia information without encrypting an entire file**

Joe J

Solution:

- **Selective Encryption**
- **Team ISE**

Joe J

Definition of Selective Encryption:

- **Selective encryption applies encryption to a subset of a file with the expectation that the entire file will be rendered useless to anyone who cannot decrypt that subset.**

Joe J

Successful Selective Encryption:

- **The right subset must be chosen or the file may not very secure**

Example:

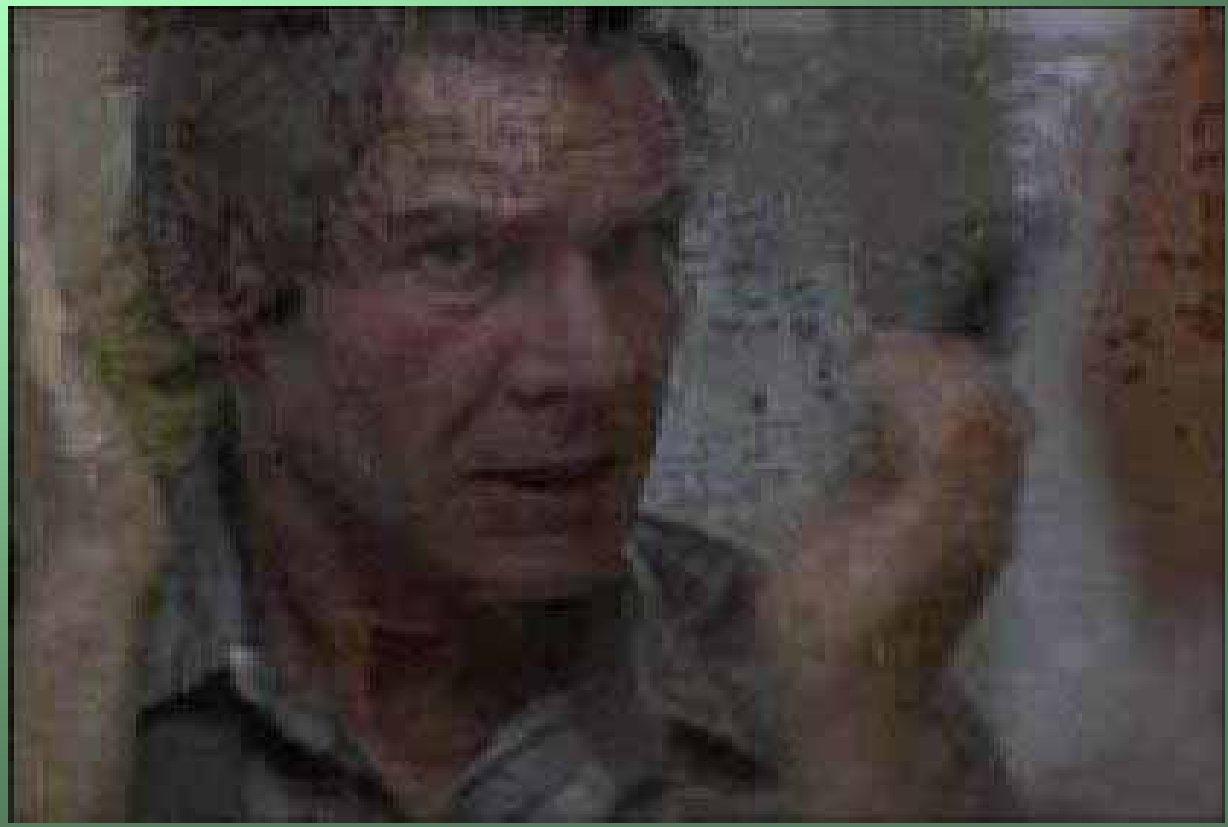
```
For ixstanxe, obsxurinx
everx fiftx lettxr of ax
Englxsh sextencx does xot
maxe it pxrticxlarlx hard
xo read.
```

Joe J

Sponsor's current research into MPEG

- Target for degraded image rather than secretive
- Encryption of less than 10% of file
- Allow only a small reduction in efficiency (nominal increase in bandwidth)
- Provide solution to the cryptography community for review and testing

Joe J



Joe J

Joe J

Team ISE Presentation Outline:

- General introduction to compressed media
- Requirements for JPEG image approach
- Architecture and Design
  - C++ Production Code
  - Website
- Demonstrate research
  - C# JPEG Manipulator

Joe J

Compressed Multimedia Overview

Shinya

Multimedia file examples:

- **MPEG 1, 2, and 4 video**
- **MP3 (MPEG 1 Layer 3)**
- **JPEG, zip and voice**

Each uses a standard compression scheme

Multimedia File Components

- **Compressed files are partitioned into standard pieces**
- **Some parts are *Descriptive***
- **Others are *Mathematical***
- **These components are referred to as frames within a compressed media file**

Shinya

Frame:

- **Consists of a marker, header and data**
- **Example frame (piece of a JPEG file):**

```
ff e0
00 10
4a 46 49 46 00 01 01 01 00 48 00 48 00 00
```

Shinya

Marker:

- **Indicates what kind of data follows.**
- **Example marker:**

ff e0 (indicates Application Data)

Shinya

Header:

- **Describes the data to follow**
- **Size, parameters, descriptor, etc.**

- **Example header:**

00 10 -- (16 bytes of data will follow)

Shinya

Data:

- **The information itself**
- **Example data:**

4a 46 49 46 00 01 01 01 00 48 00 48 00 00

(16 bytes of information indicating what application created the file.)

Shinya

Real cryptography approaches a solution from both a white hat and black hat view

- **White hat -- Designs a secure system**
- **Black hat -- Attempts to break into the system**

Closed encryption method

- **NSA has been faulted in not publishing their algorithms**
- **Only people on the inside actually know the algorithm**

Open encryption method

- **Algorithm is published publicly**
- **People not involved in producing the algorithm can review and attempt to break the encryption**
- **If the encryption is broken, we can improve the algorithm**

Shinya

Why do we want to use selective encryption?

Drawbacks to encrypting an entire file

- **Takes time to encrypt the data**
- **Sometimes makes the file bigger**

Drawbacks to selective encryption

- **Slightly more complex than encrypting an entire file**
- **Have to find the right target to encrypt**

Shinya

- **Many multimedia compression algorithms concentrate critical information in a small portion of the bit stream**

- **Encrypting this portion could render the remaining information useless**

- **Selective encryption involves selecting which pieces of a bit stream to encrypt in order to minimize the amount of encryption applied and maximize the amount of damage**

JPEG Specification and Approach

Joseph K

JPEG Selective Encryption

- JPEG is a compressed image format consisting of frames (markers, headers, data)

- Through a process of analysis, we were able to find a target appropriate for successful selective encryption

- We are only required to handle Baseline JPEG Images, the most commonly used compression mode for JPEGs

Criteria For Bad Targets

- **Optional markers**
- **Markers not used in Baseline JPEG images**
- **Markers that contain information that does not affect visibility of the image**
- **Markers that contain information that can be easily guessed or forged by a hacker**

Markers immediately eliminated:

- **APP - Application**
- **COM - Comments**
- **DAC - Define Arithmetic Conditioning Tables(Not part of Baseline Compression)**
- **DHP - Define Hierarchical Progression (Not part of Baseline Compression)**
- **DNL - Define Number of Lines**
- **DRI - Define Restart Interval**
- **EOI - End of Image**
- **EXP - Expand (Not part of Baseline Compression)**

Markers immediately eliminated:

- **JPG - Reserved for Future Extensions**
- **RES - Reserved**
- **RST - Restart (Not part of Baseline Compression)**
- **TEM - Temporary (Not used in Baseline Compression)**

- **Markers themselves are predictable**
- **Scan Header easily reconstructed**

Remaining Target List for Selective Encryption

- **Encoded Data Stream**
- **Quantizer Tables**
- **Huffman Tables**

Target Analysis

- Two C++ programs were written for target analysis:

    Convert and Analyze

    Convert
    - Binary to Hexadecimal
    - File information for a single JPEG image

Joseph K

This is an ASCII representation (in hexadecimal) of the binary values found in the file : Dust.jpg

Markers Found:===============

ff d8 -- Start of Image

ff e0 -- Application Data -- 00 10 -- (16 bytes) -- 4a 46 49 46 00 01 01 01 00 48 00 48 00 00

ff db -- Define Quantization Table -- 00 43 -- (67 bytes) -- 00 06 04 05 06 05 04 06 06 05 06 07 07 06 08 0a 10 0a 0a 09 09 0a 14 0e 0f 0c 10 17 14 18 18 17 14 16 16 1a 1d 25 1f 1a 1b 23 1c 16 16 20 2c 20 23 26 27 29 2a 29 19 1f 2d 30 2d 28 30 25 28 29 28

ff db -- Define Quantization Table -- 00 43 -- (67 bytes) -- 01 07 07 07 0a 08 0a 13 0a 0a 13 28 1a 16 1a 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28

ff c0 -- Huffman Table -- Baseline DCT -- 00 11 -- (17 bytes) -- 08 01 cb 02 4a 03 01 22 00 02 11 01 03 11 01

ff c4 -- Huffman Table -- 00 1f -- (31 bytes) -- 00 00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0a 0b

Joseph K

Analyzer

- Compute file information for multiple JPEG images
- Average file size
- Average number of Huffman tables
- Average size of Huffman tables combined
- Average number of Quantizer tables
- Average size of Quantizer tables combined

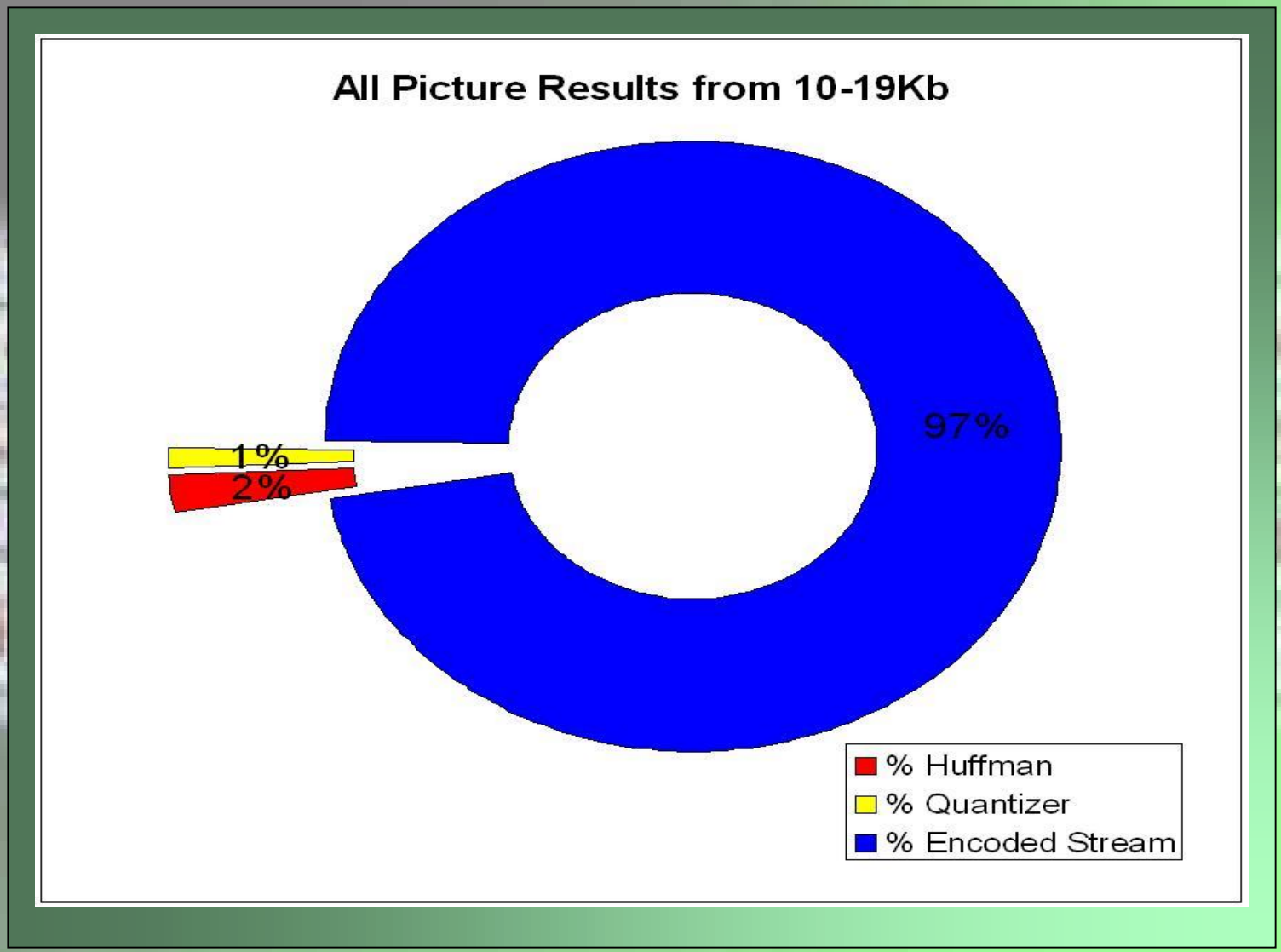
Analyzer

- Average size of the encoded stream
- Average number of markers
- Number of files processed

- Percent of the file dedicated to:
  - Huffman tables
  - Quantizer tables
  - Encoded Stream

Test Cases for JPEG Analysis

- Over 2500 JPEG images randomly selected from the Internet
- Digital Photograph vs. Manmade
- Size ranges: 10-19KB, 100 KB, 1 MB, and larger
- Resolution Ranges: 320x240, 640x480, and 800x640 Pixels


All Picture Results from 10-19Kb
97%
1%
2%
% Huffman
% Quantizer
% Encoded Stream

Joseph K

Encoded Data Stream

- SOI (start of image) marker
- Compressed data stream
- Takes up a large portion of the file
- Averaged 90% of the file!

Quantizer Tables

- DQT (Define Quantization Table) markers
- Averaged 0.88% of the file
- Unpredictable affects on image
- Might not visually damage the image at all!
- Could be replaced with another Quantizer table!

Huffman Tables

- DHT (Define Huffman Table) markers
- Averaged 1.84% of the file
- Considerable damage to image
- Mathematically derived from the image
- This makes the Huffman Tables a perfect target for Selective Encryption

Architecture and Design

Andrew

ISE ARCHITECTURE

- Invocation
- ISE Class Inheritance
- User Interface
- High-Level Modular Decomposition
- File Description

Andrew

ARCHITECTURE: INVOCATION

- Test Suite
  - Application designed to aid the research into JPEG images
  - Easy to test the effects of manipulating each type of frame
- Production Code
  - Final release of C++ package implementing selective encryption for successful targets

Andrew

ARCHITECTURE: INVOCATION

Test Suite:

- **Invoked as a windowed Version 1.1 .NET application**

Andrew

ARCHITECTURE: INVOCATION

Production Code:

- **ISE ENCRYPTOR and ISE DECRYPTOR**
  - **Input File Name and Path**
  - **Output File Name and Path**
  - **Encryption Key**
  - **Flags**

Andrew

ARCHITECTURE: ISE CLASS INHERITANCE

ISE Super Class

- The ise class will define our process and will be inherited by our specific encryption classes:

```
class ise
 {
     public:
     virtual int selectiveEncryption(. . .) = 0;
     virtual int selectiveDecryption(. . .) = 0;
};
```

Andrew

ARCHITECTURE: ISE CLASS INHERITANCE

JPEG Encryptor Class

- The jpeg_file class is the immediate goal of project ISE

```
class jpeg_file : public ise
{
    public:
    virtual int selectiveEncryption(. . .);
    virtual int selectiveDecryption(. . .);
};
```

Andrew

ARCHITECTURE: ISE CLASS INHERITANCE

MP3 Encryptor Class

- This class will be an extension to the project.

```
class mp3_file : public ise
{
  public:
    virtual int selectiveEncryption(. . .);
    virtual int selectiveDecryption(. . .);
};
```

Andrew

ARCHITECTURE: USER INTERFACE

- Test Suite
- Production Code

Andrew

ISE JPEG Manipulator - Version 1.0.1 - C:\Dust.jpg

File Help

New Project
Open Project
Save Project
Open Picture
Update Picture
Exit

Picture | Manipulated Picture

Project | File Information | Encoded Data | Quantizer Table | Huffman Tables 1 | Huffman Tables 2 | Application Data | Misc

Huffman 1: ffc0
08 01 cb 02 4a 03 01 22 00 02 11 01 03 11 01

Huffman 2: ffc4
00 00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00 00
01 02 03 04 05 0f 0f 0f 0f 0f 0f

Original 1:
Restore

Original 2: ffc4
00 00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00 00
01 02 03 04 05 06 07 08 09 0a 0b
Restore

Huffman 3: ffc4
10 00 02 01 03 03 02 04 03 05 05 04 04 00 00 01 7d 01
02 03 00 04 11 05 12 21 31 41 06 13 51 61 07 22 71 14

Huffman 4: ffc4
01 00 03 01 01 01 01 01 01 01 01 01 00 00 00 00 00 00
01 02 03 04 05 06 07 08 09 0a 0b

Original 3:
Restore

Original 4:
Restore

ARCHITECTURE: ISE USER INTERFACE

Production Code:

- Utilized through API
  - Encryptor API

```
int selectiveEncryption(
    ifstream &input_file_stream,
    ofstream &output_file_stream,
    char* key_material,
    char* encryption_flags,
    int num_flags, int quality);
```

Andrew

ARCHITECTURE: ISE USER INTERFACE

Production Code:

- **Utilized through API**
  - **Decryptor API**

```
int selectiveDecryption(
    ifstream &input_file_stream,
    ofstream &output_file_stream,
    char* key_material,
    char* encryption_flags,
    int num_flags, int quality);
```

Andrew

FILE DESCRIPTIONS

Input Files:

- **Encryptor:**
  - **JPEG file with ‘.jpg’ extension**
- **Decryptor:**
  - **Encrypted file with ‘.ise’ extension**

Andrew

FILE DESCRIPTIONS

Output Files:

- **Encryptor:**
  - **Encrypted file ending in '.ise' extension**
- **Decryptor:**
  - **JPEG file with '.jpg' extension**

Andrew

ISE WEBSITE

- Website is temporarily Available at:
  - ucsub.colorado.edu/~pouzeshi/ISE
- Includes links to:
  - Sponsor
  - Documentation
  - Test Suite/Production Code
  - Other Relevant sites
  - Team ISE info

Andrew

PROJECT EXTENSIONS/CODE MODIFICATION

Compression Type Extension:

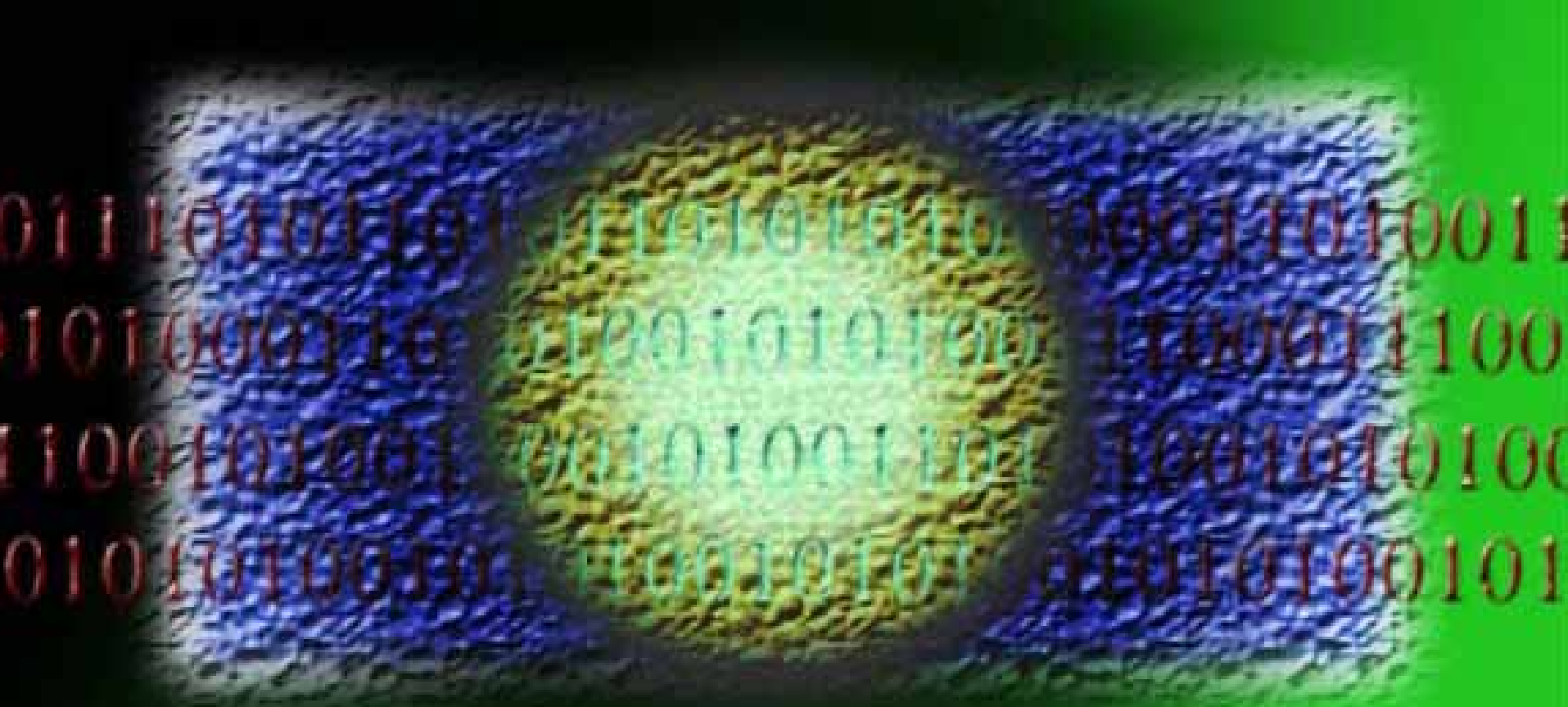
- **Time Permitting, the project will be extended to include:**
  - **MP3 Compression**
  - **ZIP Compression**

- **The Modular Design of the JPEG selective encryptor makes these extensions possible.**

Andrew

Manipulator Demonstration:

Geoff



Team ISE

Image Selective Encryption

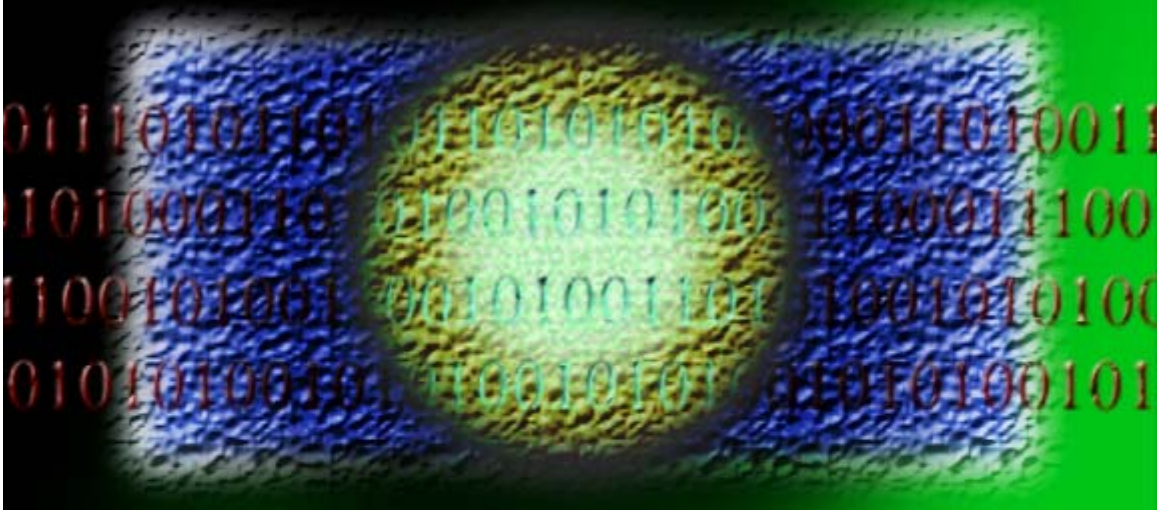
Requirements Specification

Team ISE

Image Selective Encryption

Requirement Specification

31 October 2003



Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Project Proposal

A constant amount of traffic flows between computers connected to the Internet. A large volume of information may take a long time traveling from destination to destination. The resulting speed reduction makes it desirable to compress the file as much as possible in order to send the smallest amount of data. Compression of data has allowed for the high-speed data transfers that have made Internet communication and business very workable.

In addition to sending the smallest amount of information possible, users also attempt to maintain a certain level of security upon their information. Due to the fact that common encryption methods generally manipulate an entire file, most encryption algorithms tend to make the transfer of information more costly in terms of time and bandwidth. Thus, users pay a price for security relative to their desired level of security. One possible solution would be a system of encryption that works cooperatively with the standard compression schemes. *Selective Encryption* of only a small percentage of the file's bits will facilitate this solution. Because most encryption schemes will make the file larger, selective encryption seeks only to encrypt portions of the file that will make it unusable. In other words, if a user does not have the proper decryption device, the file should not be usable. Selective encryption will minimize the necessary increase in file size due to encryption while maintaining a maximum level of uselessness, or damage, to the product.

An image could be encrypted with any of the sufficiently secure encryption algorithms available to the open source community, but this will usually result in a dramatic increase in file size that will severely increase transfer time over the Internet. However, selecting key parts of a file for encryption and only encoding those bits can actually render an image unusable. The initial statistical analysis done by the team will consist of specifically breaking down the standard JPEG compression scheme into its usable parts and evaluate which of these, if encrypted, will cause a potential user to pay for rights to the image or force subscription to the provider service.

Team ISE (Image Selective Encryption) will deliver a package for selectively encrypting JPEG (Joint Photographic Experts Group) still image files. The package will provide the tools necessary to encrypt the critical information of a JPEG file in cooperation with existing standard compression tools. This package will handle JPEG files in such a way that only a small percentage of the total file will be encrypted. Selective Encryption security will not extend to the level of military secrecy, but rather a level that would deter all but brute force attacks, allowing users to securely protect private JPEG images.

An additional aspect of the encryption analysis will be the determination of the specific targets in the file for encryption. For example in an MPEG file there are headers that contain a small portion of the overall number of bits but which are extremely vital to the reproduction of the movie by the user. So, if certain headers were to be encrypted the percentage of the file being manipulated would be less than ten percent of the total number of bits in the file. Although only a small portion will be encrypted, the resulting

damage experienced by an unauthorized user would be sufficient to cause the user to pay for the decryption package. However, there are other targets that, while they can be encrypted and will do sufficient damage, can be guessed by an attacker. The attacker could, with some degree off effort, render the file useful without use of the decryption software. For example, if the frame rate of an MPEG file was encrypted, an attacker could try all three of most common frame rates and one of these is certain to produce the correct rate for the particular video. In the case of JPEG Selective Encryption, Team ISE will have to balance the targets for encryption against ease of simple attacks.

A permanent website will be constructed by the team to make the software package available to anyone interested in the software process. As it is vital to the world of cryptography to let the community view the approach, the first form of the working prototype will be made available on the website. From this, feedback can be received not only from the team itself, but also from the cryptography community at large.

So, following the guidelines of the ongoing MPEG research (also being guided by the sponsor), the team will study the JPEG process and earlier attempts at encryption. With the sponsor's assistance, Team ISE will devise a workable approach to handling individual JPEG images following the concept of selective encryption.

It is possible that the team will complete the JPEG process early enough in the year that they will able to apply the same approach to other types of compressed files (text, audio, etc.) However, this initial specifications document applies only to the envisioned JPEG project.

Table of Contents

1. INTRODUCTION

Team ISE is being sponsored by Assistant Professor of Computer Science, Tom Lookabaugh, at the University of Colorado: http://itd.colorado.edu/lookabaugh/. Tom Lookabaugh is currently involved in selective encryption research on standard MPEG (Moving Picture Experts Group) files and is interested in researching the application of Selective Encryption for other multimedia formats.

The goal of selective encryption is to minimize the amount of encryption applied to a file while maximizing the damage done to the image being viewed by a user not in possession of the authorized decryption package. Complete encryption is not a requirement of the process, nor is rendering the file useless to the level of complete military secrecy. It is acceptable for an attacker to be able to view portions of the file; however, the file should be distorted enough that an attacker would not wish to use the encrypted file, but would rather purchase or subscribe to the decryption method for access to the original files.

Multimedia files prove to be a good subject for selective encryption, as these files tend to be very large and employ compression algorithms that concentrate critical information in small portions of their bit stream. If the critical data in certain multimedia standards is encrypted properly, the remaining information becomes useless to those without the appropriate decrypter. There are many types of compression algorithms that fit this description, such as MPEG 1, 2 and 4 video, G.723 and G.729 video, AAC audio, MP3 audio, JPEG and JPEG2000 image formats. Applying a Selective Encryption security solution to selected multimedia formats will greatly increase the protection level of important information.

The focus of the ISE project is to research and develop an algorithm for selectively encrypting the JPEG *baseline* compression image standard. The product of the research and development will be a package that will encrypt a file so that the amount of the file being encrypted is relatively small (on the order of 1-2% of the total file). The product will be delivered in a package that will include an encrypter and a decrypter for JPEG files, a website to facilitate the delivery of the product and documentation about the process. The encrypter and decrypter will encrypt and decrypted selected targets contained within JPEG files. The ISE project will employ the AES (Advanced Encryption Standard) for our Selective Encryption algorithm. This package will be made available in a purely open source form on our final website.

In addition to the package containing the decrypter and encrypter, Team ISE will also provide a test suite available to prospective users. The test suite will be used to aid in the research, development and testing of the team's final product. The test suite will provide the functions necessary to completing this project. First, it will allow the user to preview a standard JPEG image. Second, the test suite break down the various portions of a JPEG image and provide the ability to manipulate the data of all of the pieces the particular file. Third, after altering the data in any particular file, the test suite will provide the capability

to preview the encryption attempt without the benefit of compatible decryption. Forth, the suite will have the ability to decrypt an encrypted file. The decryption options will allow the user try to defeat the encryption methods (let the user put on a black hat). Any selective encryption scheme could be developed using a package that implemented these features, however, the delivered test suite will only employ the AES encryption scheme chosen by the team. The test suite will be available to download from the team website.

The final website will be deployed on a sponsor provided Apache web servers. The machine facilitating the web server will use the Linux Red Hat 9.0 operating system platform. The team will acquire a fixed IP address from the proper University of Colorado authorities and will develop a simple website capable of delivering information to viewers about the benefits and application of Selective Encryption technology. The site will provide users the option to download and use the final software package. The site will also provide links to important information and will remain in place as long as the sponsor deems necessary.

The envisioned software package will accomplish the complex task of selectively encrypting a JPEG baseline standard image, while providing a simple user interface to users. Team ISE has identified three specific types of users: high-end art users, typical Internet image users, and small, low-end image users. The research and software will be tailored to these users' needs. Figure 1.1 is a flow chart showing the general logic design of the team's final product.

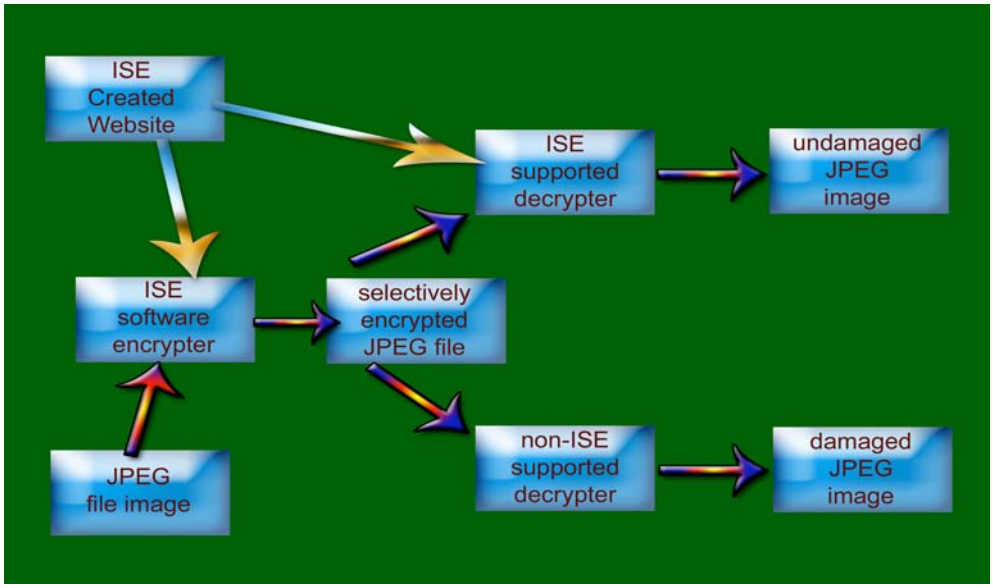


Figure 1.1: Conceptual Overview of ISE Software

Information regarding the research required by the sponsor is further outlined in the next section. Details regarding the requirements of Team ISE's Selective Encryption project are then presented, followed by short discussions of possible requirements alternatives and future enhancements. These details are concluded with a summary followed by a glossary of important terms and a list of related readings.

2. RESEARCH PATH

2.1. Research and Analysis Requirements

The research and analysis will be the initial part of this project. The final product of the research will allow the team to determine a specific approach to this form of Selective Encryption. The sponsor considers the research done by the team equally as important as the delivery of the final product. The research portion of this project will include:

- A proportional analysis of a large quantity of JPEG images to define what might be acceptable targets within the JPEG file structure for encryption.
- An analysis of earlier methods of encryption for performance and effectiveness.
- The analysis of potential encryption methods and targets in the JPEG image file for percentage of file encryption vs. image corruption.
- The analysis of different encryption methods and targets in the JPEG image file for the encryption target's susceptibility to potential attack.
- The final stage of the research analysis will conclude with approval from the sponsor on the useful approaches and corresponding performance issues.

2.2. Research Related Products

Following significant discoveries throughout the project the Sponsor requires that the team will produce research documentation to be presented at applicable security and compression conferences.

3. REQUIREMENTS

The requirements have been divided into several sections based upon the category of the requirement. These categories consist of the Supporting Environment, Functional, Performance, Documentation and Release Requirements. Each of the requirements is defined below.

➢ 3.1. Supporting Environment

The supporting environment includes specification of both the expected environments that the package should be able to perform in and the form in which the package will be written. There is also a specification of the minimum hardware environment the package will require to be run on. The package referred to in the requirements consists of the encryption and decryption package. The test suite is not a portion of this package, and has its own runtime and language requirements.

- **3.1.1. Software**

  The supporting software environment includes the runtime environment as well as the development requirements.

  - **3.1.1.1. Runtime Environment**
    - ❖ Package to be operational in Linux Red Hat 9.0, Windows XP and Mac OS X.
    - ❖ The test suite is to be operational in a .NET environment.
    - ❖ Web page is to be viewable on Internet Explorer 6.0 and Safari 1.0.

  - **3.1.1.2. Development Environment**
    - ❖ Package to be written in ANSI C/C++ specification.
    - ❖ Package should not change IJG's claim of wide portability (see http://www.ijg.org for specific environments).
    - ❖ CVS will be used for software versioning.
    - ❖ Test suite to be written in the C# (C-sharp) programming language.
    - ❖ The web page will be built on a server utilizing Linux Red Hat 9.0 operating system, supplied by the sponsor.
      - ✓ Web page will use HTML version 4.01.

- **3.1.2. Hardware**
  - Package should be able to be run on any computer system supporting color graphics.
  - Generic color monitor and JPEG image viewing system outlined above.
  - Keyboard as part of the user interface.
  - Hardware supports the software environment outlined above.

➢ 3.2. Functional Requirements

The functional requirements specify all of the functionality Team ISE's product is required to provide. These requirements will include the interface to our production code and outline the functionality that must be supported. The requirements of the test suite and web page will also be listed in this section.

- **3.2.1. Required Operations**
  - Encrypt a selected portion of a JPEG baseline standard image compression format.
  - Decrypted files must maintain compliance to the JPEG compression standard.
  - Level of encryption is not "secretive/military" but level to provide sufficient protect against all but brute force attacks.

- **3.2.2. Package Functionality**
  - Must be able to read in .jpg or files for encryption.
    - ❖ Encryption software must gracefully terminate upon receipt of other file types.
  - Must output a file ending with an .ise extension. The .ise extension denotes a Team ISE selectively encrypted file.
  - Decrypt module will input and process .ise file types.
    - ❖ Decryption software must gracefully terminate upon receipt of other file types.
  - Decrypt will output a standard .jpg file.
  - Final product will be a software package that provides simple interface methods.

- **3.2.3. ISE Function Interfaces**

  This section illustrates the pseudo code form that the Team ISE Selective Encryption package interface methods must have.
  - int selectiveEncryption(ifstream &input_file_stream, ofstream &output_file_stream, char* key_material, char* encrypt_flags, int num_flags, int quality);
  - int selectiveDecryption(ifstream &input_file_stream, ofstream &output_file_stream, char* key_material, char* decrypt_flags, int num_flags, int quality);

- **3.2.4. Test Suite Requirements**
  - **3.2.4.1 Test Suite Display**
    - The test suite must have a standard windows display.
    - The test suite must include buttons and tabs to display different portions of the JPEG image file data.
    - Figure 3.2.4.1 displays a screenshot of the test suite.

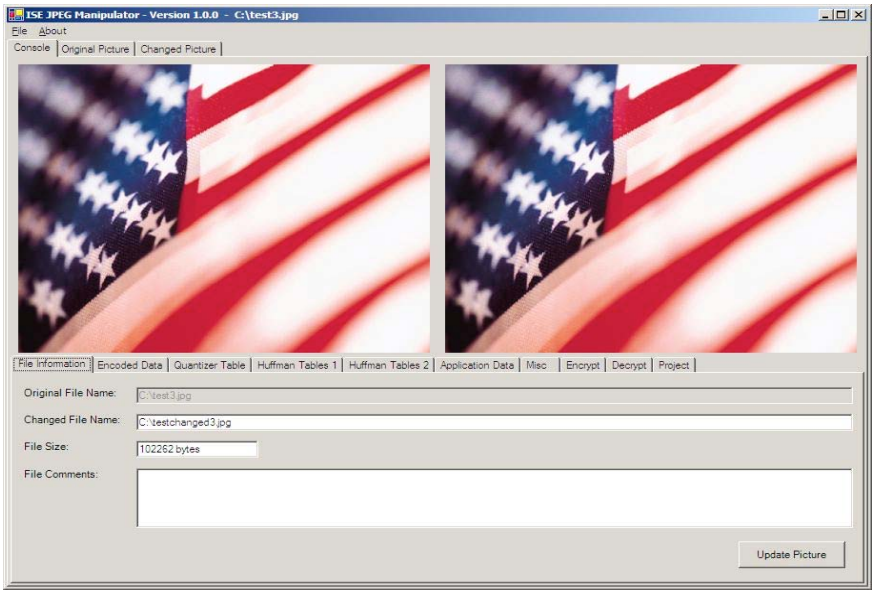


Figure 3.2.4.1 Test Suite Screen Shot.

  - **3.2.4.2 Test Suite Functionality**
    - The test suite must be able to parse compressed JPEG files.
    - The test suite must divide up and display the hexadecimal values of the different pieces of a JPEG file.
    - Display the manipulated image of the file alongside the original image. The image size will be altered to fill the display windows.
    - The test suite must allow the user to make changes to the data contained in each of the pieces of a JPEG file.
      - ✓ These changes must be incorporated into the encoding of the file, and the image displayed must be updated to be the image produced by the changes in the encoding.
    - Include tabs to display the image without altering its size.
    - Incorporate the encryption and decryption software methods provided in the final software product.
      - ✓ The test suite will provide a graphical user interface for the final Team ISE product package.

- Provide options for the user to save all of the information and changes to the current image.

- **3.2.5 Supporting Web Page**
  - To be deployed on machine provided by the sponsor.
  - Contain links explaining the purpose of the software package provided by Team ISE.
  - Contain links to downloadable version of the software package.
  - Contain links to downloadable version of the test suit package.
  - Contain links to the software documentation as well as providing the user with the ability to download the documentation.
  - Contain open source files of the software package.
  - Contain links to other sources of related information.
  - Contain information about the sponsor.
  - Contain information about Team ISE.

3.3. Documentation and Release Requirements

The following requirements specify the documentation that is to be provided, along with issues related to the release and delivery of the final product.

- **3.3.1. Documentation Requirements**
  - Man Page - A standard UNIX man page.
  - User Tutorial - A presentation of system for first-time user.
  - Research paper(s) written in a style specified by the sponsor.
  - A website to include all code and documentation and supporting links.
  - Documents will be made available in Adobe PDF file format.

- **3.3.2. Release Requirements**
  - Product will be delivered as series of ZIP files for Unix, Windows and Mac users.
  - Files will include installation programs for automatic generation and installation of executable and preview/evaluation programs.
  - Documentation will be provided on the website for download.

4. Future Enhancements

Pending the early completion of the JPEG selective encryption methods and software, the following enhancements may be incorporated into the final product. These enhancements illustrate the development of selective encryption, and its spread into other areas in the future.

➢ 4.1 Selective Encryption of MP3 Files

The project may be extended to include the MP3 file format. The team will research MP3 file formats and devise ways of selectively encrypting MP3 files. Completion of this will entail expanding the encrypter/decrypter software to include MP3 files. The software will output selectively encrypted MP3 files. For security and consistency, these encrypted files will have the same ".ise" extension as the encrypted JPEG files. The test suite will also be expanded to parse MP3 files and display the encoding to the user. Documentation will be updated to include descriptions of the MP3 encoder/decoder. Research papers involving MP3 selective encryption will be produced upon the Sponsor's request. The website will be updated to include all pertinent MP3 information.

➢ 4.2 Selective Encryption of ZIP Files

Upon completion of both the JPEG and MP3 selective encryption targets, the project will be further extended to the ZIP file format. Team ISE will update the encrypter/decrypter software to perform upon ZIP file types. Again, for security and consistency, selectively encrypted ZIP files will end in the ".ise" extension. Research will be done to determine how to best perform selective encryption upon ZIP files. The test suite will again be expanded to parse ZIP files and display the encoding to the user. Documentation will be updated to include descriptions of the ZIP encoder/decoder. Research papers involving ZIP selective encryption will be produced upon the Sponsor's request. The website will be updated to include all pertinent ZIP information.

5. SUMMARY

The purpose of this document was to give an outline for the path of research and the set of requirements for the ISE software package. These requirements include the software and hardware environments the application will run on, the functional requirements of the software, test suite and website, the research and analytic requirements, and the supporting and research document's requirements that will be included along with the software package. This document includes all necessary information for designing all of the necessary aspects of the Team ISE software.

6. Glossary

AES (Advanced Encryption Standard)
An encryption method that uses block ciphering.

ANSI C/C++
The standard C and C++ programming languages as defined by the American National Standards Institute.

Black Hat
The process of testing an encryption algorithm by trying to break the encryption using several different methods.

Baseline JPEG
A subset mode of sequential JPEG where the number of tables is restricted and the sample precision must be eight bits.

C#
A modern, object-oriented language that enables programmers to quickly build a wide range of applications for the new Microsoft .NET platform.

Compression Algorithm
An algorithm designed to compress a file, that is, utilizes patterns in a file to reduce the size of the file.

CVS (Concurrent Versioning System)
A code management system. CVS provides the ability to track (and potentially revert) incremental changes to files, reporting them to a mailing list as they are made, and can be used concurrently by many developers.

Decryption
The act of rendering an encrypted file into a know format.

Encryption
To convert computer data or messages to something incomprehensible by means of a key, so that only an authorized recipient holding the matching key can recover the original.

IJG (Independent JPEG Group)
An informal group that writes and distributes a widely used free library for JPEG image compression. IJG is not affiliated with the ISO committee.
www.ijg.org/

ISO (International Organization for Standardization)

The world's largest developer of standards, particularly the development of technical standard.

JPEG (Joint Photographic Experts Group)

A compression technique for color images and photographs that balances compression against loss of detail in the image. The greater the compression, the more information is lost (this is called Lossy compression).

Military Secrecy

A level of secrecy where all information is hidden.

MP3 (MPEG-1 Audio Layer-3)

A standard technology and format for compression a sound sequence into a very small file (about one-twelfth the size of the original file) while preserving the original level of sound quality when it is played.

MPEG (Moving Picture Experts Group)

A standard for digital video and audio compression.

Selective Encryption

A method of encryption that exploits the relationship between encryption and compression to reduce encryption requirements, saving in complexity and facilitating new system functionality. Selective Encryption only encrypts a small portion of a file.

Visual Studio .NET

Microsoft's visual programming environment for creating web services based on use of the Extensible Markup Language (XML).

ZIP

A method of compressing text files.

7. Related Readings

[Lookabaugh and Sicker and Keaton and Gua and Vedula 2003]

Lookabaugh, T., and Sicker, D., and Keation, D., and Guo, W., and Vedula, I. *Security Analysis of Selectively Encrypted MPEG-e Streams*. 2003.

Tom Lookabaugh's description of the methods and results of applying selective encryption to MPEG-2 streams.

[Miano 99]

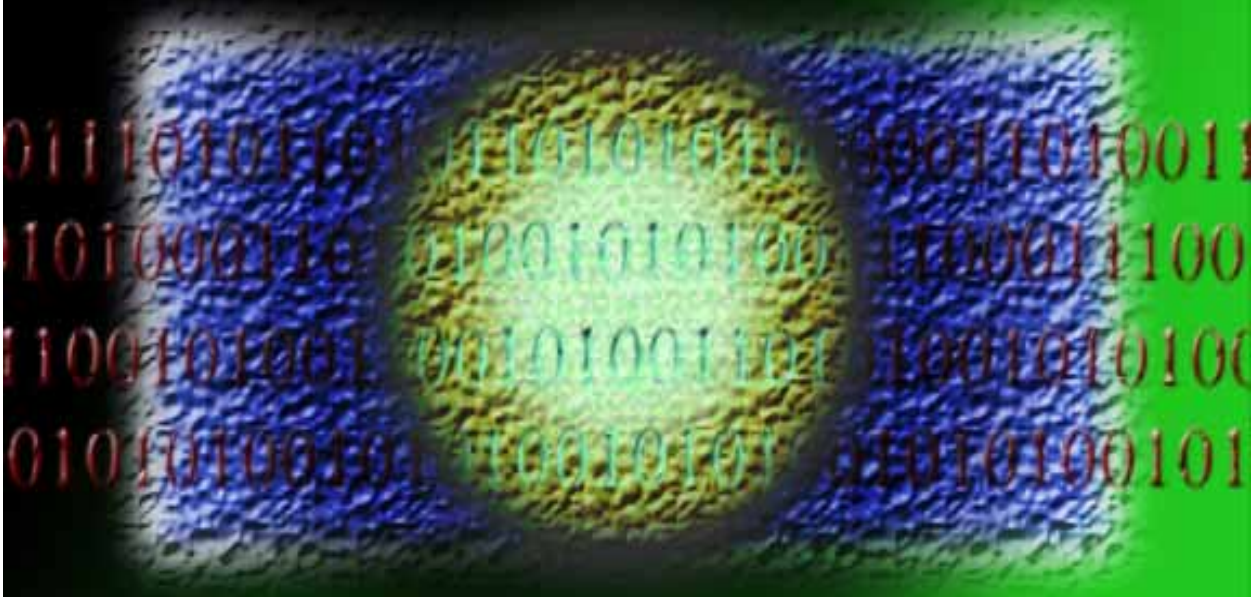
Miano, J. *Compressed Image File Formats*. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

Design Specification

Team ISE

Image Selective Encryption

Design Specification
5 December 2003



Team ISE
Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Project Proposal

Traffic constantly flows between computers connected to the Internet. Large volumes of information may take a long time traveling from destination to destination. Such a reduction in speed makes it desirable to compress the file as much as possible in order to send the smallest amount of data required. Thus, compression of data has allowed for the high-speed data transfers that have made Internet communication and business more feasible.

In addition to sending the smallest amount of information possible, users also attempt to maintain a certain level of security upon their information. Due to the fact that common encryption methods generally manipulate an entire file, most encryption algorithms tend to make the transfer of information more costly in terms of time and bandwidth. Thus, users pay a price for security relative to their desired level of security. One possible solution would be a system of encryption that works cooperatively with the standard compression schemes. *Selective Encryption* of only a small percentage of the file's bits will facilitate this solution. Because most encryption schemes will make the file larger, selective encryption seeks only to encrypt portions of the file that will make it unusable. In other words, if a user does not have the proper decryption device, the file should not be usable. Selective encryption will minimize the necessary increase in file size due to encryption while maintaining a maximum level of uselessness, or damage, to the product.

Team ISE (Image Selective Encryption) will deliver a package for selectively encrypting JPEG (Joint Photographic Experts Group) still image files. The package will provide the tools necessary to encrypt the critical information of a JPEG file in cooperation with existing standard compression tools. This package will handle JPEG files in such a way that only a small percentage of the total file will be encrypted. Selective Encryption security will not extend to the level of complete encryption, but rather to a level that would deter all but brute force attacks, allowing users to securely protect private JPEG images.

A JPEG image could be encrypted with any of the sufficiently secure encryption algorithms available to the open source community, but this can result in an increase in file size or can require a large amount of processing time. However, by selecting small but vital portions of a file and encrypting only those few bytes can render an image unusable. The initial statistical analysis done by the team will consist of specifically breaking down the standard JPEG compression scheme into its usable parts and evaluate which of the parts, if encrypted, will cause a potential user to pay for rights to the image or force subscription to the provider service.

An additional aspect of the encryption analysis will be the determination of the specific targets in the file for encryption. For example in an MPEG file there are headers that contain a small portion of the overall number of bits but which are extremely vital to the reproduction of the movie by the user. So, if certain headers were to be encrypted the percentage of the file being manipulated would be less than ten percent of the total number of bits in the file. Although only a small portion will be encrypted, the resulting damage experienced by an unauthorized user would be sufficient to cause the user to pay for the decryption package. However, there are other targets that, while they can be encrypted and will do sufficient damage, can be guessed by an

attacker. The attacker could, with some degree of effort, render the file useful without use of the decryption software. For example, if the frame rate of an MPEG file was encrypted, an attacker could try all three of most common frame rates and one of these is certain to produce the correct rate for the particular video. In the case of JPEG Selective Encryption, Team ISE will have to balance the targets for encryption against ease of simple attacks.

A permanent web site will be constructed by the team to make the software package available to anyone interested in the Team's project. As it is vital to the world of cryptography to let the community view the approach, the first form of the working prototype will be made available on the web site. From this, feedback can be received not only from the team itself, but also from the cryptography community at large.

So, following the guidelines of the ongoing MPEG research (also being guided by the sponsor), the team will study the JPEG process and earlier attempts at encryption. With the sponsor's assistance, Team ISE will devise a workable approach to handling individual JPEG images following the concept of selective encryption.

It is possible that the team will complete the JPEG process early enough in the year that they will able to apply the same approach to other types of compressed files (text, audio, etc.). However, this specifications document applies only to the envisioned JPEG project

Table of Contents

1. INTRODUCTION

Team ISE is sponsored by Assistant Professor of Computer Science, Tom Lookabaugh, at the University of Colorado: http://itd.colorado.edu/lookabaugh/. Tom Lookabaugh is currently involved in selective encryption research on standard MPEG (Moving Picture Experts Group) files and is interested in researching the application of Selective Encryption for other multimedia formats.

The goal of selective encryption is to minimize the amount of encryption applied to a file while maximizing the damage done to the image being viewed by a user not in possession of the authorized decryption package. Complete encryption is not a requirement of the process, nor is rendering the file useless to the level of complete military secrecy. It is acceptable for an attacker to be able to view portions of the file; however, the file should be distorted enough that an attacker would not wish to use the encrypted file, but would rather purchase or subscribe to the decryption method for access to the original files.

Multimedia files prove to be good subjects for selective encryption, as these files tend to be very large and employ compression algorithms that concentrate critical information in small portions of their bit stream. If the critical data in certain multimedia standards is encrypted properly, the remaining information becomes useless to those without the appropriate decryptor. There are many types of compression algorithms that fit this description, such as MPEG 1, 2 and 4 video, G.723 and G.729 video, AAC audio, MP3 audio, JPEG and JPEG2000 image formats. Applying a Selective Encryption security solution to selected multimedia formats will greatly increase the protection level of important information.

The focus of the ISE project is to research and develop an algorithm for selectively encrypting the JPEG *baseline* compression image standard. The product of the research and development will be a package that will encrypt a file so that the amount of the file being encrypted is relatively small (on the order of 1-2% of the total file). The product will be delivered in a package that will include an encryptor and a decryptor for JPEG files and a testing suite. A web site will be constructed to facilitate the delivery of the product and documentation about the process. The encryptor and decryptor will encrypt and decrypt selected targets contained within JPEG files. The ISE project will employ the AES (Advanced Encryption Standard) for our Selective Encryption algorithm. This package will be made available in a purely open source form on our final web site.

In addition to the package containing the decryptor and encryptor, Team ISE will also provide a test suite available to prospective users. The test suite will be used to aid in the research, development and testing of the team's final product. The test suite will provide the functions necessary to complete this project. First, it will allow the user to preview a standard JPEG image. Second, the test suite will break down the various portions of a JPEG image and provide the ability to manipulate the data in all of the portions. Third, after altering the data in any particular file, the test suite will provide the capability to preview the encryption attempt without the benefit of compatible decryption. Forth, the suite will have the ability to decrypt an encrypted file. The decryption options will allow the user try to defeat the encryption methods. Any selective encryption scheme could be developed using a package that implemented these

features, however, the delivered test suite will only employ the AES encryption scheme chosen by the team. The test suite will be available to download from the team web site.

The final web site will be deployed on a web server provided by the Sponsor. The machine facilitating the web server will use the Linux Red Hat 9.0 operating system platform. The team will acquire a fixed IP address from the proper University of Colorado authorities and will develop a simple web site capable of delivering information to viewers about the benefits and application of Selective Encryption technology. The site will provide users the option to download and use the final software package. The site will also provide links to important information and will remain in place as long as the sponsor deems necessary.

The final software package will accomplish the complex task of selectively encrypting a JPEG baseline standard image while providing a simple user interface. Team ISE has identified three specific types of users: high-end art users, typical Internet image users, and small, low-end image users. The research and software will be tailored to these users' needs. Figure 1.1 is a flow chart showing the general logic design of the team's final product.

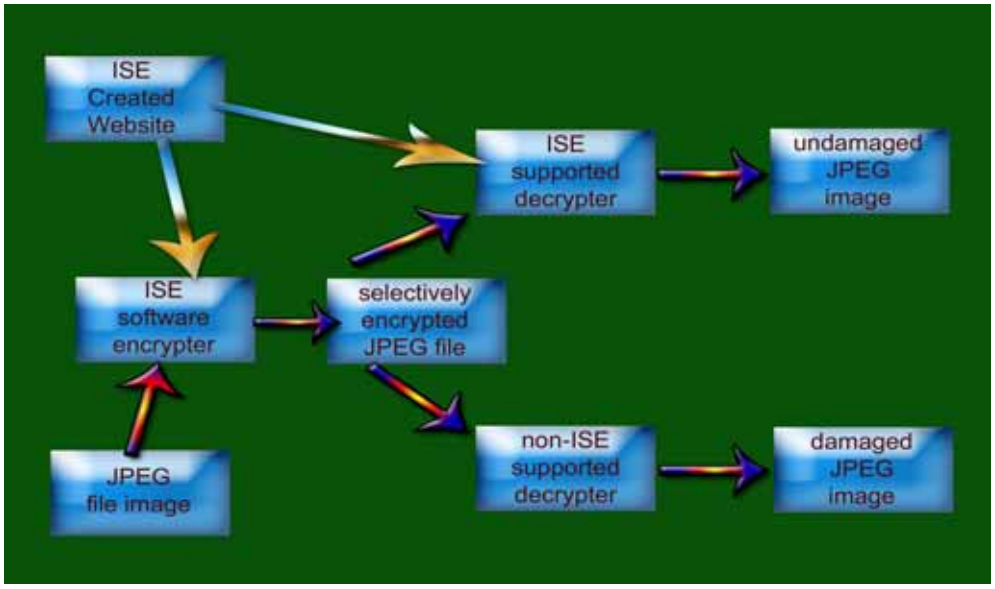


Figure 1.1: Conceptual Overview of ISE Software

Information regarding the user interfaces for all of the ISE products are described in the next section of this document. Following the user interface sections, the design overview for the project is presented with a high-level modular decomposition of each of the project modules and sub-modules. After the design overview, an in-depth explanation of the design and its low-level functionality for each module is presented. Immediately following the low-level design is an explanation of all of the valid file types used by the ISE products. Lastly, a summary of this document is provided, followed by a complete glossary of terms, and finally, a listing of readings directly related to this project. For a full description of the project requirements or the system architecture document, please refer to the online documentation located on the ISE web site at http://128.138.75.184. This document outlines the full design of the ISE project and will be referred to as a "road map" for development during the implementation process.

2. USER INTERFACE

During the course of this project, Team ISE will develop three separate products: the ISE class production code, the JPEG Manipulator test suite and the Team ISE web site. Each of these products will have a different user interface. This section outlines the design of each of the three final products.

2.1. ISE Class Production Code User Interface

The user interface to the ISE production code is a series of C++ classes designed for use in application development. A software developer can create a new instance of the jpeg_ise type, input the pertinent information, and then make calls to the class APIs to encrypt and decrypt images. This section defines how a programmer may employ this functionality.

2.1.1. Instantiation of the JPEG ISE Class

To create a new instance of this class, the user may choose from three different constructors. The default constructor allows the user to create the object without having to pass any arguments. An example of using the default constructor is shown below:

```
// Creating a JPEG ISE object with default ctor
jpeg_ise MyEncryptionClass;
```

In addition to the default constructor, the jpeg_ise class also provides two overloaded constructors for passing the Key, and one (or optionally both) of the file names. If third parameters are not passed, a default name will be created based upon the input file name. An example of using the two overloaded constructors is shown below:

```
// Creating JPEG ISE objects with overloaded ctors
char * KEY_STRING = "Some Password";
char JPEG_File_Name [256] = "C:\\MyImage.jpg";
char ISE_File_Name [256] = "C:\\MyImage.ise";
char Out_File_Name [256] = "C:\\MyImageDecrypted.jpg";
jpeg_ise My_Encryption_Class(KEY_STRING, JPEG_File_Name,
                             ISE_File_Name);
jpeg_ise My_Decryption_Class(KEY_STRING, ISE_File_Name,
                             Out_File_Name);
```

2.1.2. Usage of the JPEG ISE Class Methods

Once an instance of the class has been declared, the user can then begin to make calls to the various functions. There are two major uses of the class: encrypting and decrypting. This section outlines the steps necessary to complete both of these tasks.

2.1.2.1. Encrypting with the JPEG ISE Class

There are number of steps required before the user can call the encrypt_file() method to encrypt a JPEG image. The programmer is required to set up both a key and the JPEG input file name. If desired, the user may also specify the file name for the intermediate ISE file created during the encryption process, but if none is specified, a default file name will be created based upon the original JPEG input file name. The following is an example of one way a user can encrypt using the jpeg_ise class:

```
// Objects needed to encrypt a JPEG file
jpeg_ise MyEncryptClass;
char JPEG_File_Name [256] = "C:\\MyImage.jpg";
char ISE_File_Name [256] = "C:\\MyImage.ise";

// The Key can be up to 320 chars long
char My_Key_Password [320] = "MyPassword123";

// Set the file names and key information
MyEncryptClass.set_input_file_name(JPEG_File_Name);
MyEncryptClass.set_ise_file_name(ISE_File_Name);
MyEncryptClass.set_key(My_Key_Password);

// Encrypt the JPEG file
MyEncryptClass.encrypt_file();
```

Note: This is one way in which a programmer can use the encrypt methods, but there are several ways to accomplish this task from using this class. We will talk about the design of all of the methods in section 5.1 of this document.

2.1.2.2. Decrypting with the JPEG ISE Class

The decryption process is virtually identical to the encryption process, with a few subtle differences. There are still a number of steps required before the user can call the decrypt_file() method to decrypt an ISE file. The programmer is required to set up both a key and the ISE intermediate file name. If desired, the user may also specify the name for the decrypted file, but if none is specified, a default file name will be created based upon the ISE file name. The following is an example of one way a user can decrypt using the jpeg_ise class:

```
// Objects needed to encrypt a JPEG file
jpeg_ise MyEncryptClass;
char Output_File_Name [256] = "C:\\MyDecryptedImage.jpg";
char ISE_File_Name [256] = "C:\\MyImage.ise";

// The Key can be up to 320 chars long
// The Key must match the key used to encrypt the file
char My_Key_Password [320] = "MyPassword123";

// Set the file names and key information
MyEncryptClass.set_ise_file_name(ISE_File_Name);
MyEncryptClass.set_output_file_name(Output_File_Name);
MyEncryptClass.set_key(My_Key_Password);

// Decrypt the ISE file
MyEncryptClass.encrypt_file();
```

Note: As with encrypt, this is only one way in which a programmer can use the decrypt methods, but there are several ways to accomplish this task using this class. We will talk about the design of all of the methods in section 5.1 of this document.

2.2. The JPEG Manipulator User Interface

The JPEG Manipulator's user interface is outlined within this section. The Manipulator provides an easy-to-use graphical user interface. The GUI allows the user to view all of the various pieces of a JPEG image with a familiar Windows style application interface. The following is a description of the GUI interface that will be developed for the JPEG Manipulator.

2.2.1. JPEG Manipulator Invocation

The Manipulator will come prepackaged with a fully functional installation script to provide ease of use for any user to quickly install. This package will also include an uninstaller script, to provide the user with the ability to fully remove all data installed with the program, should the need arise. Once the program has been installed by the user, they can then invoke the program from their start menu by choosing:

Start -> Programs -> ISE -> JPEG -> JPEG Manipulator

Once the user has invoked the application, it will open to the default main screen, which is the "Console" tab within the application. This will appear as a standard Windows user interface as shown below:

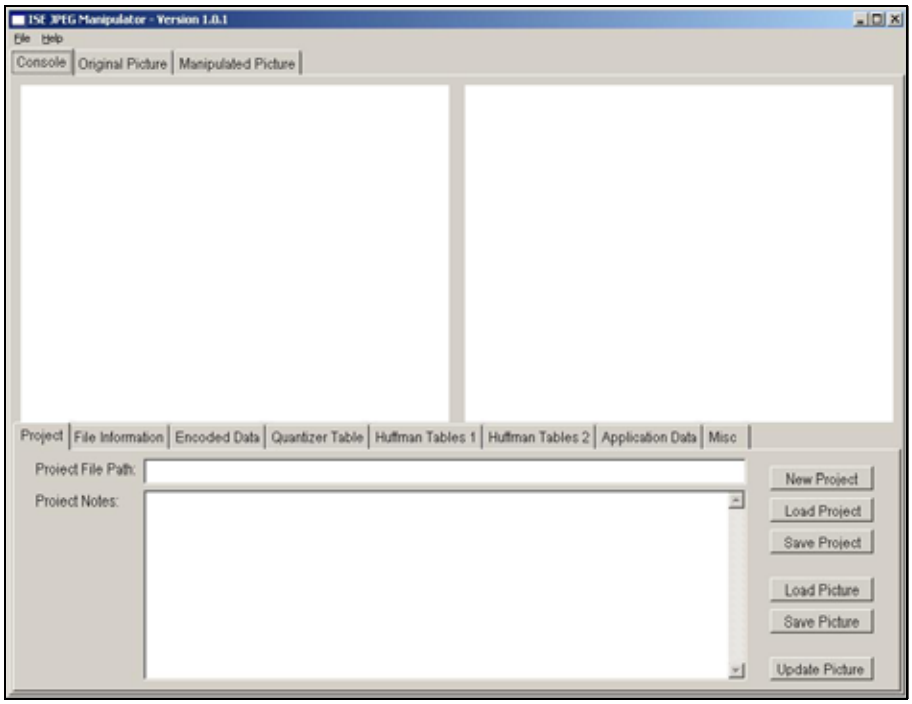


Figure 2.2.1: The JPEG Manipulator on entry into the application.

2.2.2. The Manipulator's Menu Bar

The Manipulator will provide a standard menu bar with the application. This menu bar will consist of a File, an Edit and a Help menu.

2.2.2.1. The File menu

The File menu will provide the user with a number of standard functions for interacting with the application. Figure 2.2.2.1 shows an example of the File menu. The functionality of this menu is described below.

Figure 2.2.2.1: Illustration of the File menu.

2.2.2.1.1. New Project option

This option allows the user to create a new selective encryption project within the Manipulator.

2.2.2.1.2. Open Project option

This option allows the user to open a previously created selective encryption project within the Manipulator.

2.2.2.1.3. Save Project option

This option allows the user to save the current selective encryption project that is currently in progress within the Manipulator.

2.2.2.1.4. Open Picture option

This option allows the user to open a new original JPEG image within the Manipulator.

2.2.2.1.5. Update Picture option

This option allows the user to create a manipulated image JPEG image within the Manipulator.

2.2.2.1.6. Exit option

This option allows the user to quickly and easily exit the Manipulator. If there is an unsaved project open, then the user will be prompted to save before the application has exited.

2.2.2.2. The Edit menu

The Edit menu will provide the user with the ability to do common editing functions such as Cut, Copy, and Paste. The functionality of this menu is described below.

2.2.2.2.1. Cut option

This option allows the user to cut selected text from any of the TextBox fields within the manipulator. The cut text will be copied to the system clipboard for future retrieval.

2.2.2.2.2. Copy Option

This option allows the user to copy selected text from any of the TextBox fields within the manipulator. The copied text will be copied to the system clipboard for future retrieval.

2.2.2.2.3. Paste Option

This option allows the user to paste the most recently copied or cut text from the system clipboard to the selected text box.

2.2.2.3. The Help menu

The Help menu will provide the user with a Help option and an About option. Figure 2.2.2.2 shows an example of the Help menu. The functionality of this menu is described below.

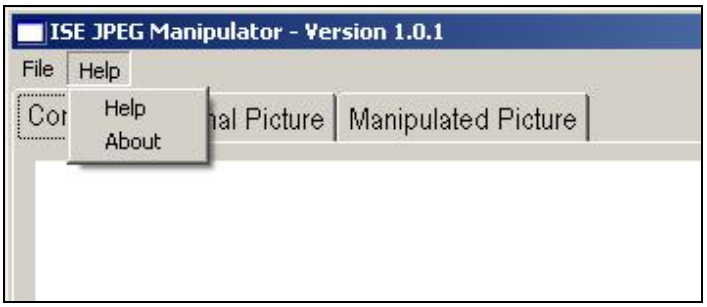


Figure 2.2.2.2: Illustration of the Help menu.

2.2.2.3.1. Help option

This option allows the user to enter the self-help portion of the program. This program will provide the user with a user-guide for the Manipulator and other information about using this program.

2.2.2.3.2. About option

This option allows the user to view the About screen included with the Manipulator. The About screen will display information about the creators, version and other minor information about the program.

2.2.3. The Manipulator's Console Tab

The Console Tab of the Manipulator consists of several different controls. The Console Tab is the main work area within the application and provides access to all of the data stored for a particular JPEG image. This access is provided via a series of Data Tabs located on the bottom half of the Console Tab. In addition, the Console Tab provides two picture box controls to view both an original image and a manipulated image, for a side-by-side comparison of the two pictures. The original picture is located on the left side and the manipulated picture is located on the right. The figure below illustrates an example of the Console Tab:

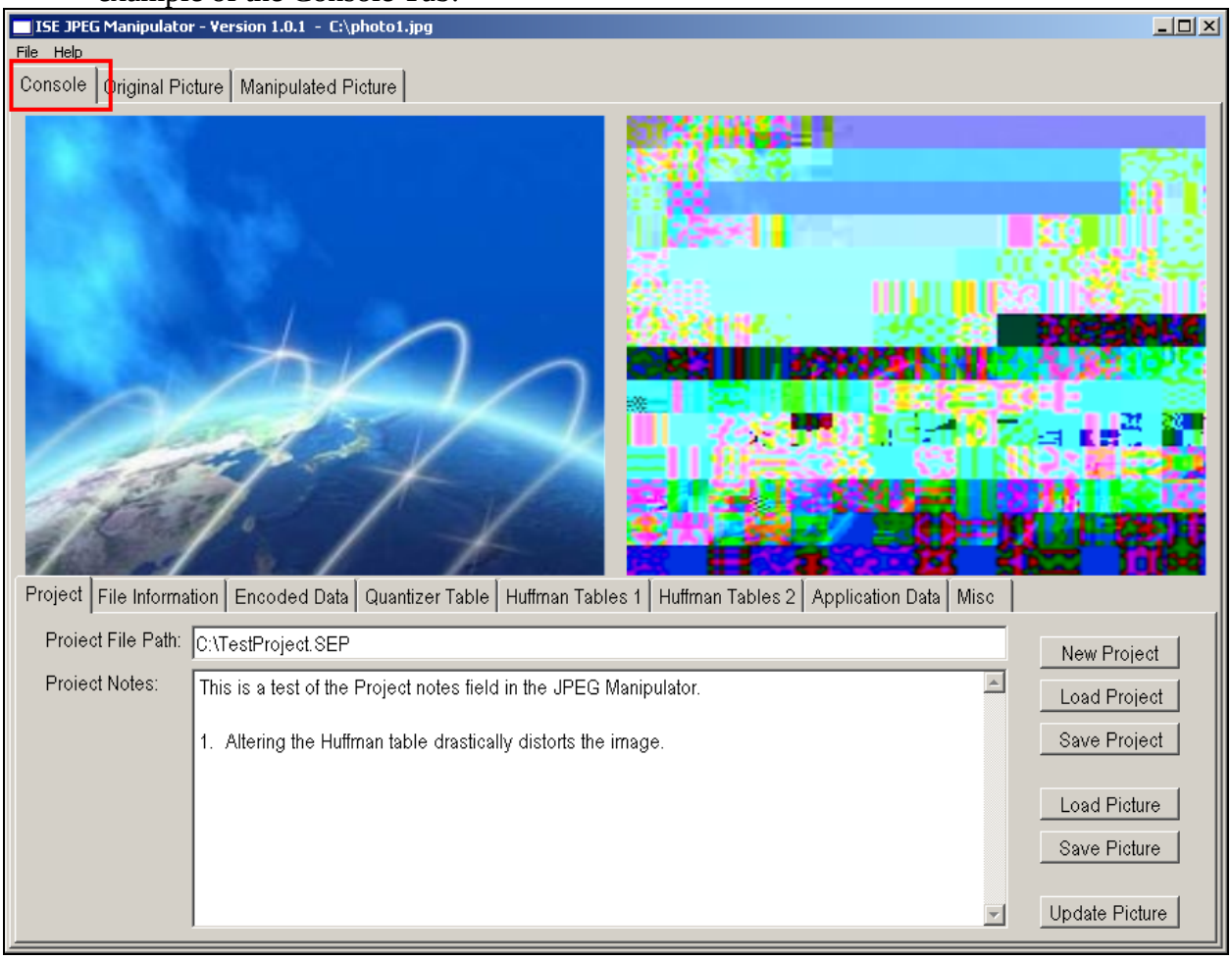


Figure 2.2.3: Example of the Console Tab

2.2.3.1. Project Tab

This tab allows the user to access project data for the currently loaded project. From here, the user can create a new project, save a project, load a project, load a picture, save a picture, create a manipulated picture or enter in notes about the particular project. The figure below shows an example of this data tab:

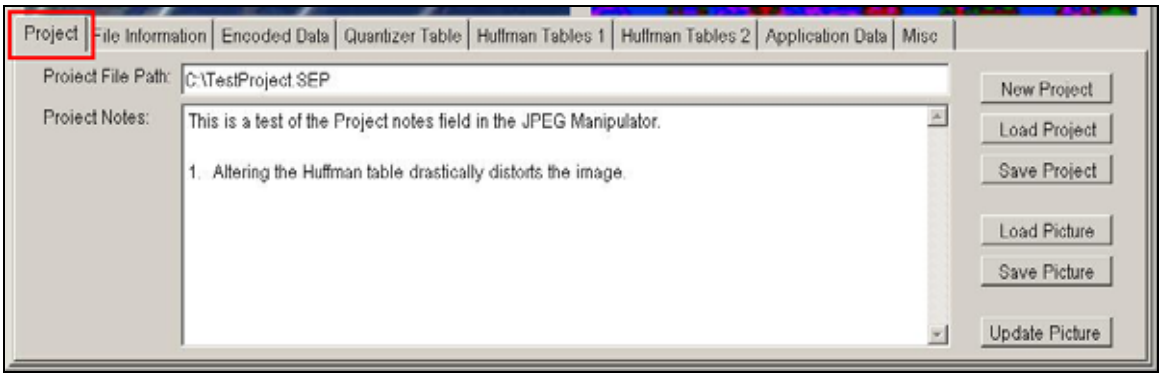


Figure 2.2.3.1: Example of the Project Tab

2.2.3.2. File Information Tab

This tab allows the user access to the File Information data for the currently loaded pictures. From here, the user can specify the manipulated picture name and path, view the original picture name and path, view the file size of the original image and view file comments included with the JPEG image. The figure below shows an example of this data tab:

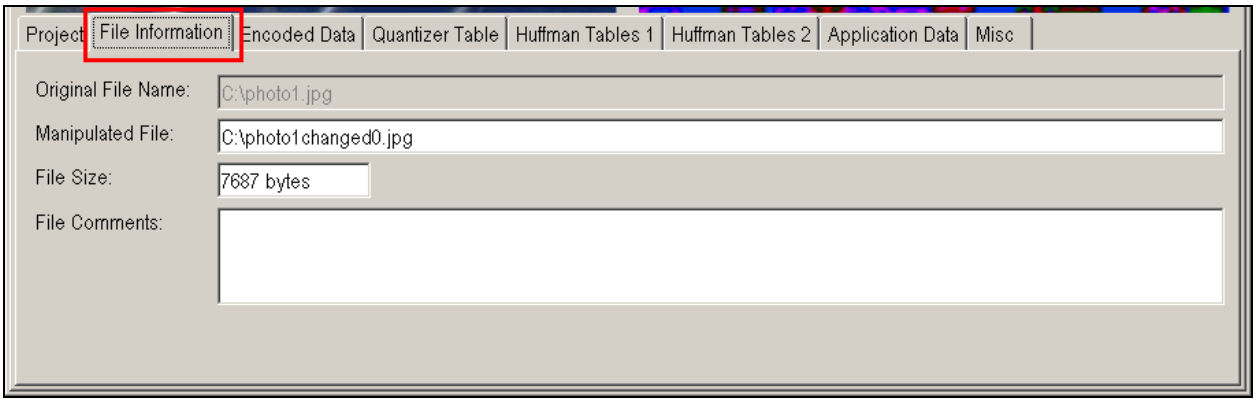


Figure 2.2.3.2: Example of the File Information Tab.

2.2.3.3. Encoded Data Tab

This tab allows the user access to the Encoded Data information for the currently loaded JPEG image. From here, the user can view and manipulate the first 10,000 bytes of the encoded data frame and the Scan Header information for the JPEG file. All of the data under this tab is displayed in hexadecimal format. The figure below shows an example of this data tab:

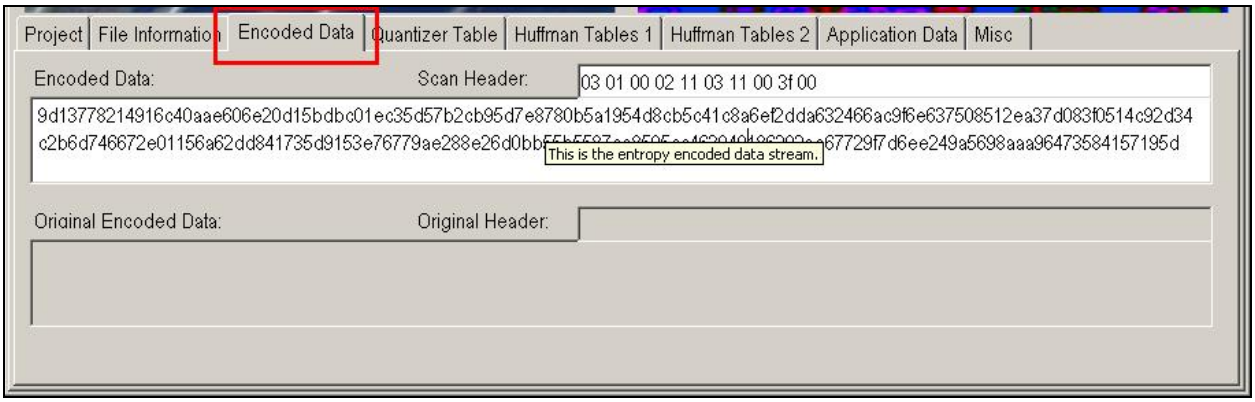


Figure 2.2.3.3: Example of the Encoded Data Tab.

2.2.3.4. Quantizer Table Tab

This tab allows the user access to the Quantizer Frame data for the currently loaded JPEG image. From here, the user can view and manipulate the Quantizer tables and restore any manipulated table to their original values. All of the data under this tab is displayed in hexadecimal format. The figure below shows an example of this data tab:

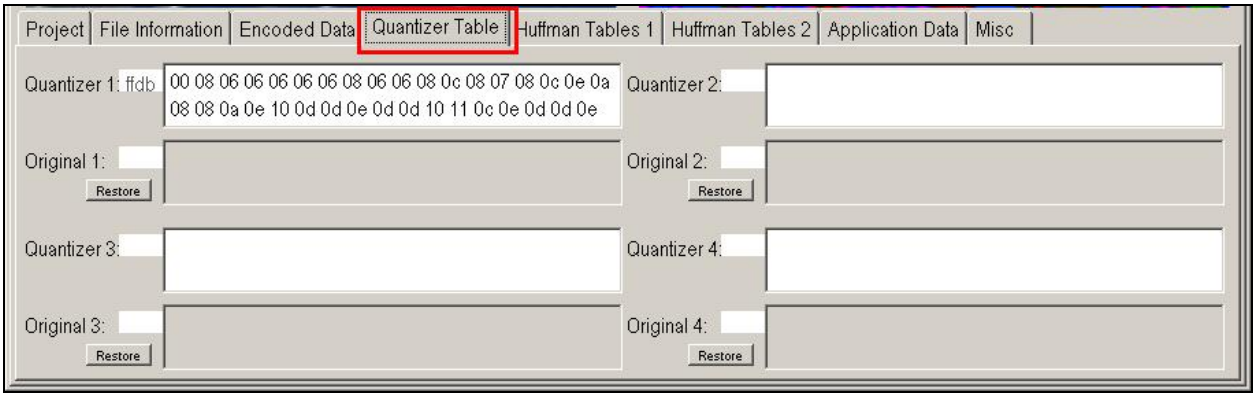


Figure 2.2.3.4: Example of the Quantizer Table Tab.

2.2.3.5. Huffman Table Tabs

This tab allows the user access to the Huffman Frame data for the currently loaded JPEG image. From here, the user can view and manipulate the Huffman tables and restore any manipulated table to their original values. Because there are up to eight Huffman tables allowed in a JPEG file, we require two tabs to present all of the data. All of the data under this tab is displayed in hexadecimal format. The figure below shows an example of one of the Huffman tabs:

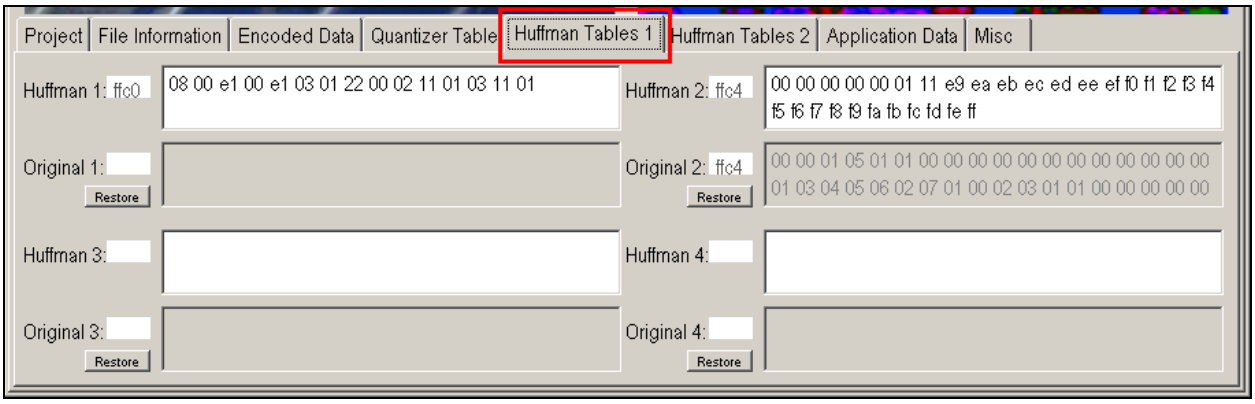


Figure 2.2.3.5: Example of a Huffman Table Tab.

2.2.3.6. Application Data Tab

This tab allows the user access to the Application Data for the currently loaded JPEG image. From here, the user can view and manipulate the Application Data contained within the image, even though this data will not affect the visual display of the JPEG image. All of the data under this tab is displayed in hexadecimal format. The figure below shows an example of the Application Data Tab:

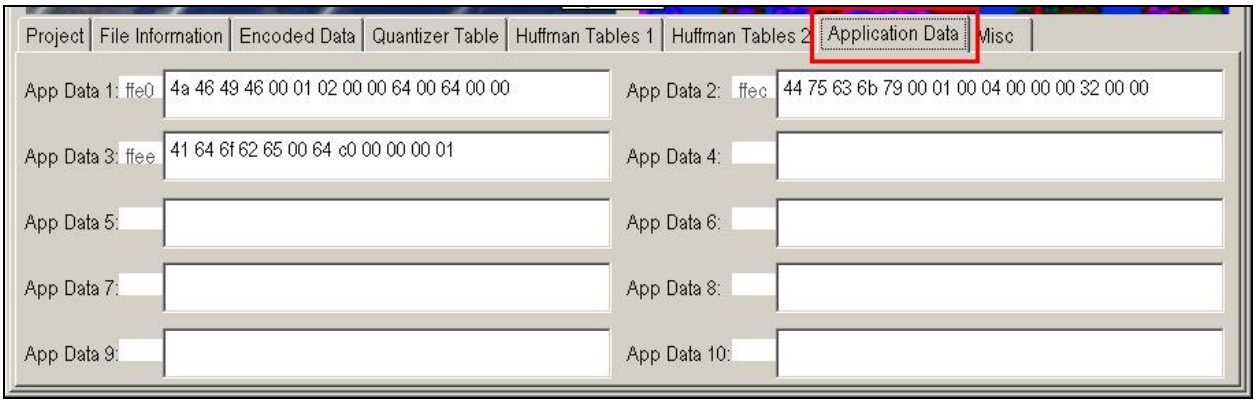


Figure 2.2.3.6: Example of the Application Data Tab.

2.2.3.7. Misc Data Tab

This tab allows the user access to the all other JPEG file data for the currently loaded JPEG image. From here, the user can view and manipulate the data for a number of data frames including: the Restart Interval, the Number of Lines, the Expand Image, the Restart Mod 8 and the data for the Hierarchical Progression. In addition to this data, the user can view any errors encountered during the use of the program. All of the data under this tab is displayed in hexadecimal format, except for the Program Errors data. The figure below shows an example of the Miscellaneous Data Tab:

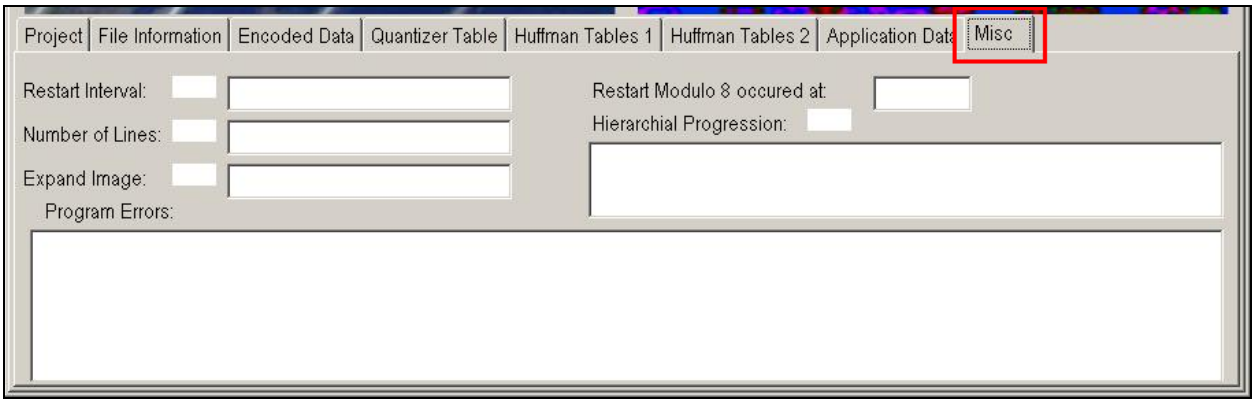


Figure 2.2.3.7: Example of the Misc Data Tab.

This sums up all of the data tabs contained under the JPEG Manipulator Console Tab. Note that all of the JPEG image data is contained within the tabs are located on the Console Tab. In addition to the Console Tab, there are two other tabs: the Original Picture Tab and the Manipulated Picture Tab. These additional tabs allow the user to view each picture in a larger form.

2.2.4. The Original Picture Tab

The Original Picture tab allows the user to view the currently loaded original JPEG image in a larger form. This tab is located directly to the right of the Console tab. The figure below illustrates an example of the Original Picture Tab:

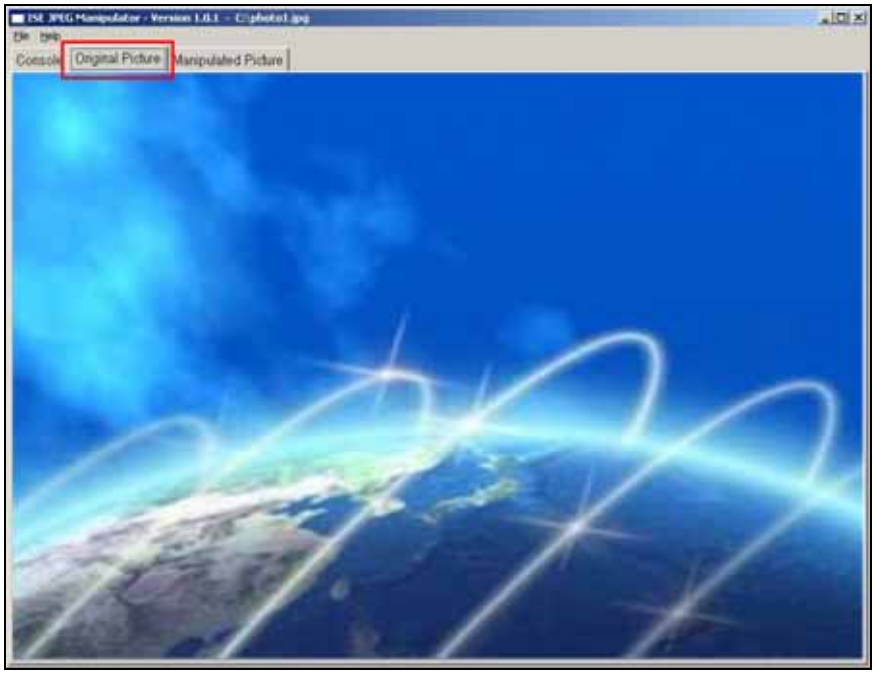


Figure 2.2.4: Example of the Original Picture Tab.

2.2.5. The Manipulated Picture Tab

The Manipulated Picture tab allows the user to view the currently loaded manipulated JPEG image in a larger form. This tab is located directly to the right of the Original Picture tab. The figure below illustrates an example of the Manipulated Picture Tab:

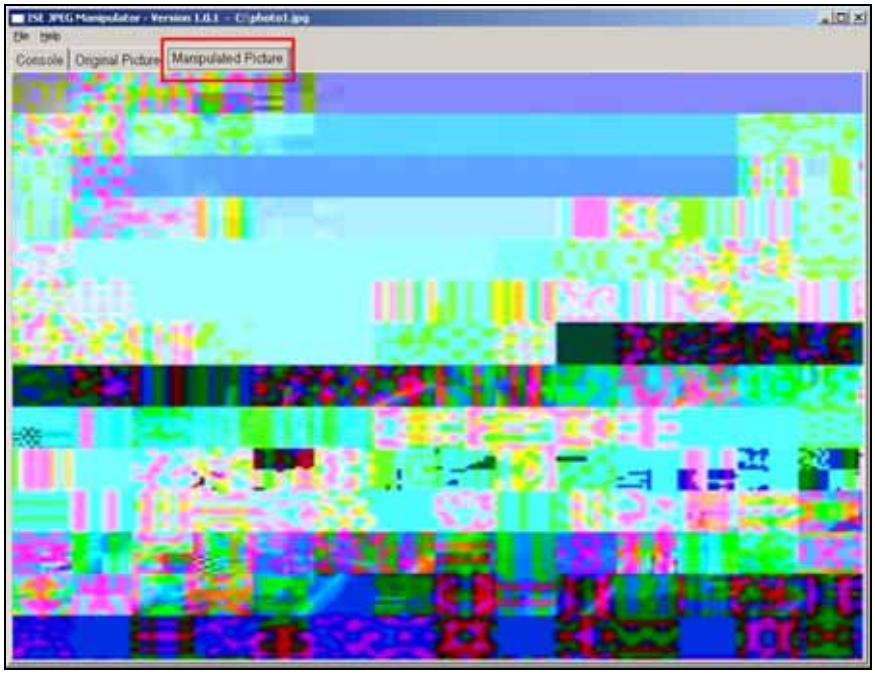


Figure 2.2.5: Example of the Manipulated Picture Tab.

2.3. Team ISE Web Site User Interface

This section outlines the user interface of the Team ISE web site. The web site will be a very simple construction with a home page directing users to previews, final product code, all final documentation and JPEG Manipulator test suite.

2.3.1. ISE Web Site Invocation

Once the project has been completed, all pertinent information will reside on the ISE web pages. This web site will be located on a server maintained by the sponsor at the University of Colorado at Boulder. The IP address of this web site is: 128.138.75.184. A user can access this site by going to their web browser and entering the following in their browser address bar:

http://128.138.75.184/

2.3.2. ISE Web Site Navigation

The user will have access to a menu at the top of the page. The buttons in the menu will redirect the user to the different sections of the web site. Users can jump directly to a specific document by using the menu's pull down menus. Figure 2.3.2.1 displays an image of the web site.

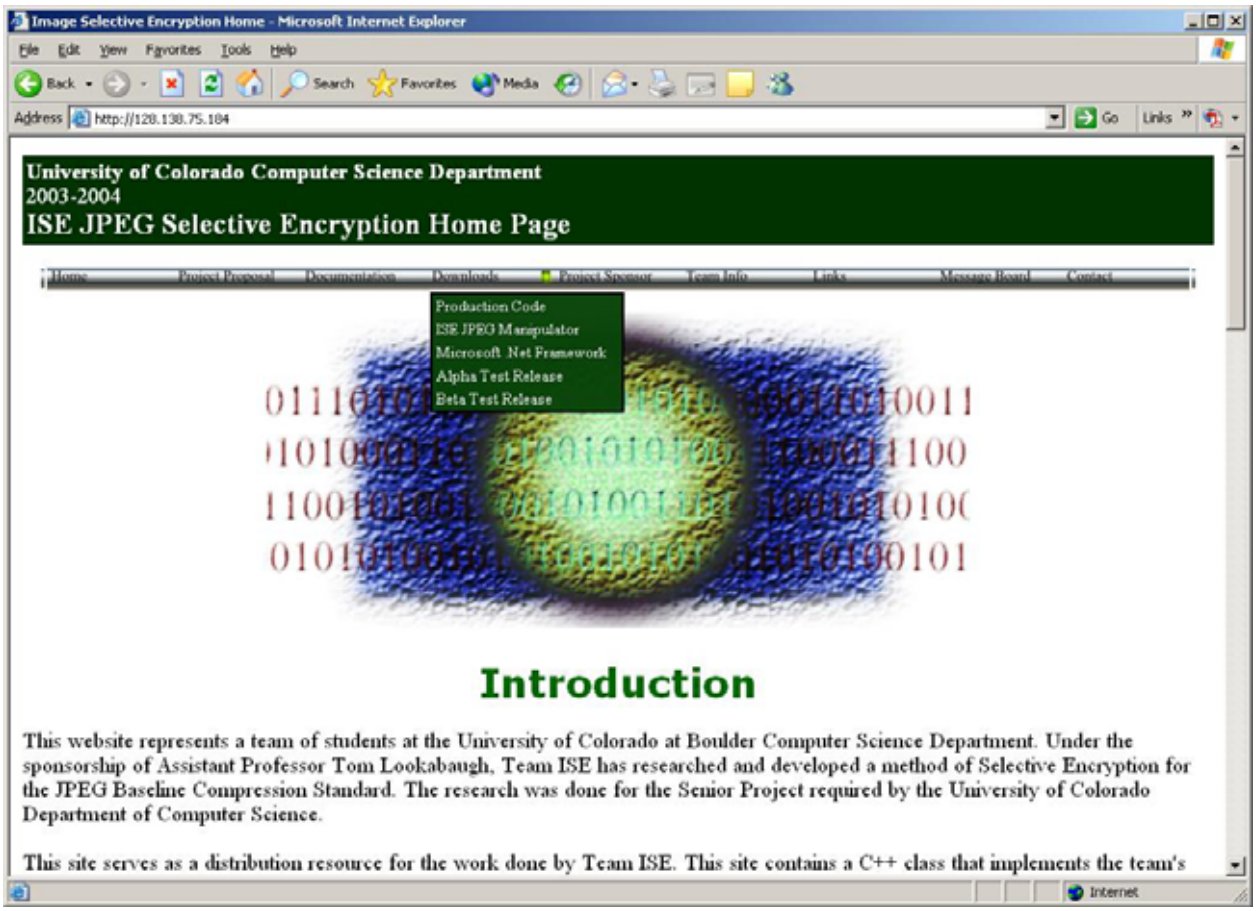


Figure 2.3.2.1: Screenshot of ISE Web Page

The Documentation button directs the user to a page where they can download the PDF version of any documents produced by Team ISE. The user can download a desired document by clicking on the document's button. Figure 2.3.2.2 displays an image of the document download page.

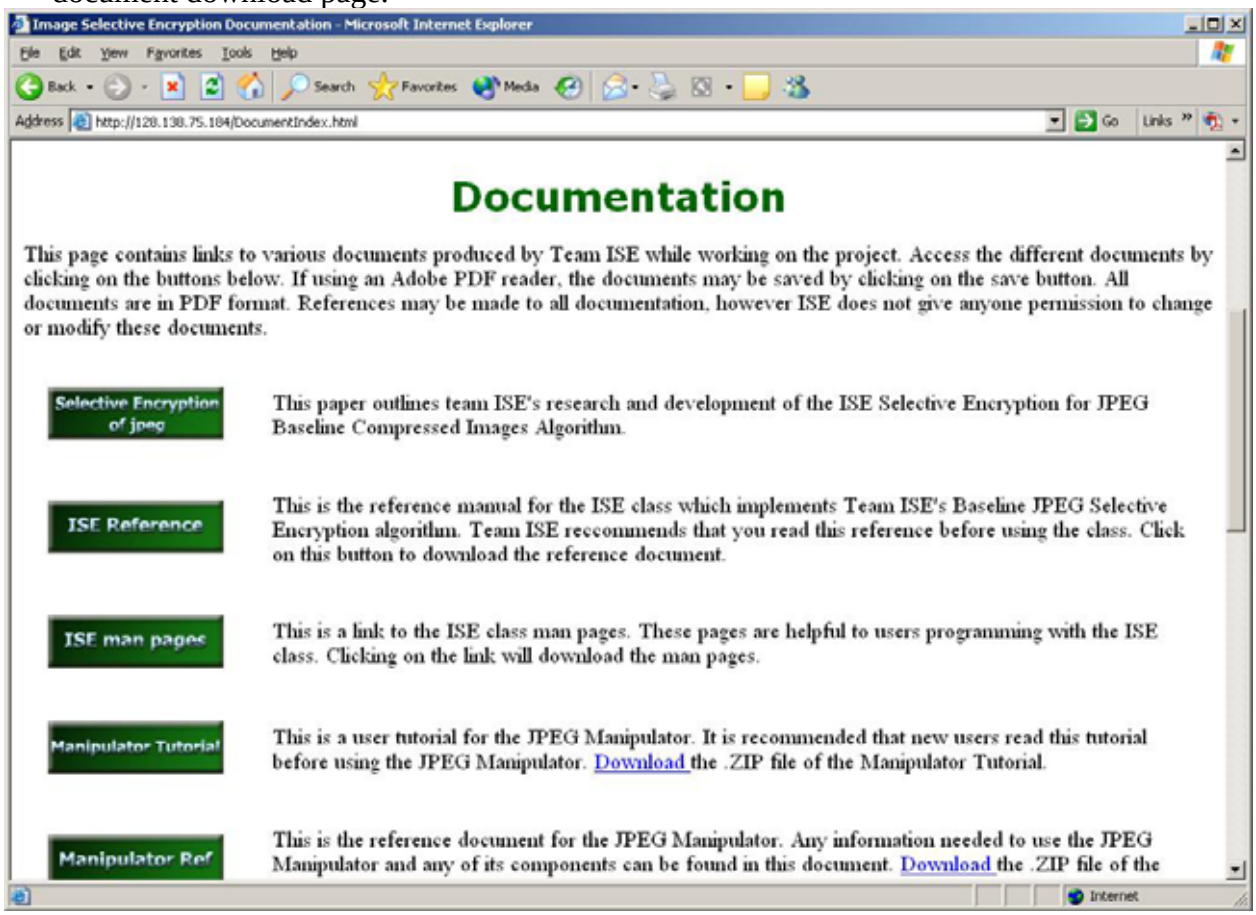


Figure 2.3.2.2: Screenshot of Documentation Page

The user can access the download page by clicking on the Download button in the menu. Upon clicking this button, the user will be directed to the download page where they can download the production code, the Manipulator, and the Microsoft .NET Framework version 1.1. The user can download these products by clicking on the buttons on the download page. Figure 2.3.2.3 displays an image of the download page.

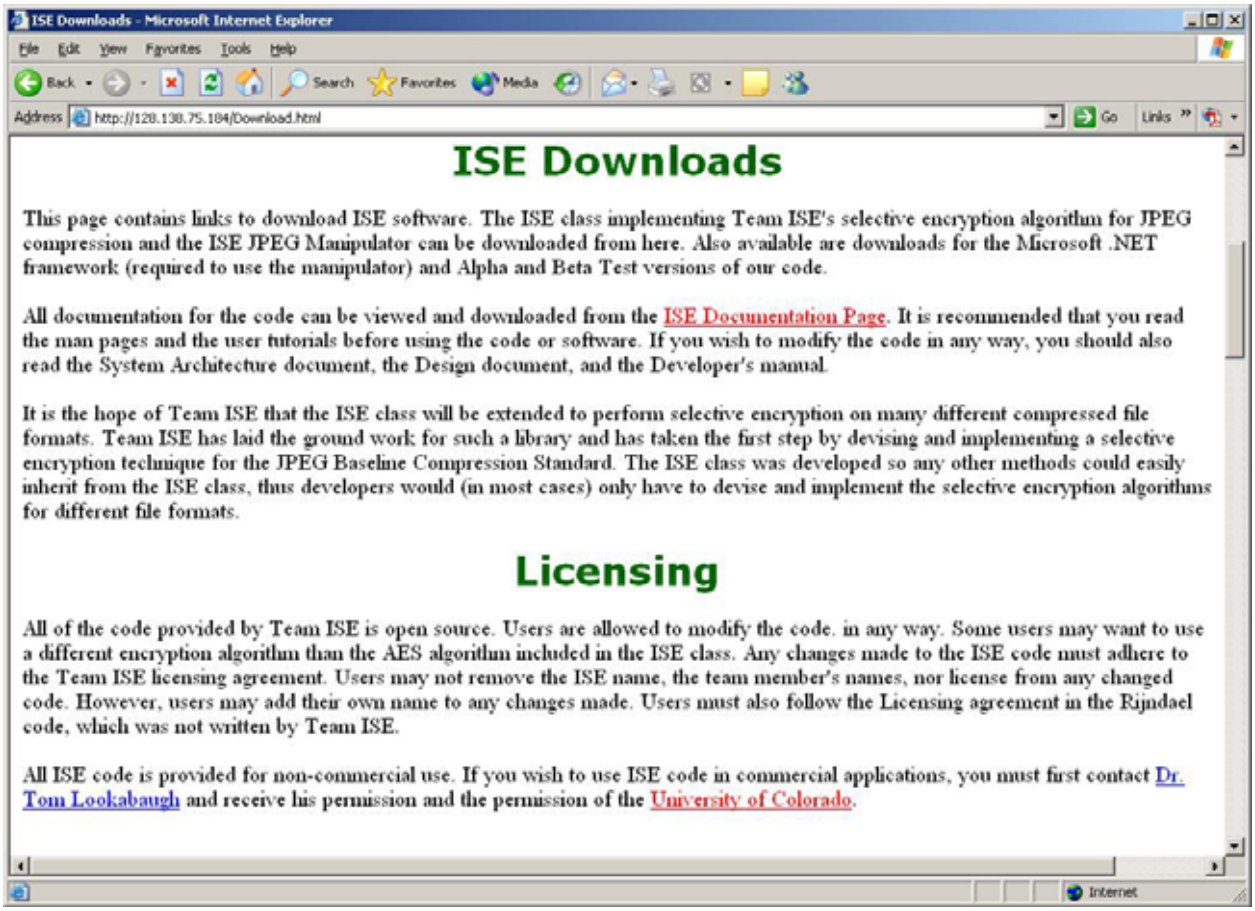


Figure 2.3.2.3: Screenshot of Download Page

The user can access relevant links by clicking on the Links button in the menu bar. The Links button will direct the user to a page containing links to web pages relevant to the ISE project. The user can visit these pages by clicking on the buttons on the links page. These links will redirect the user to other web pages. Figure 2.3.2.4 displays an image of the links page.

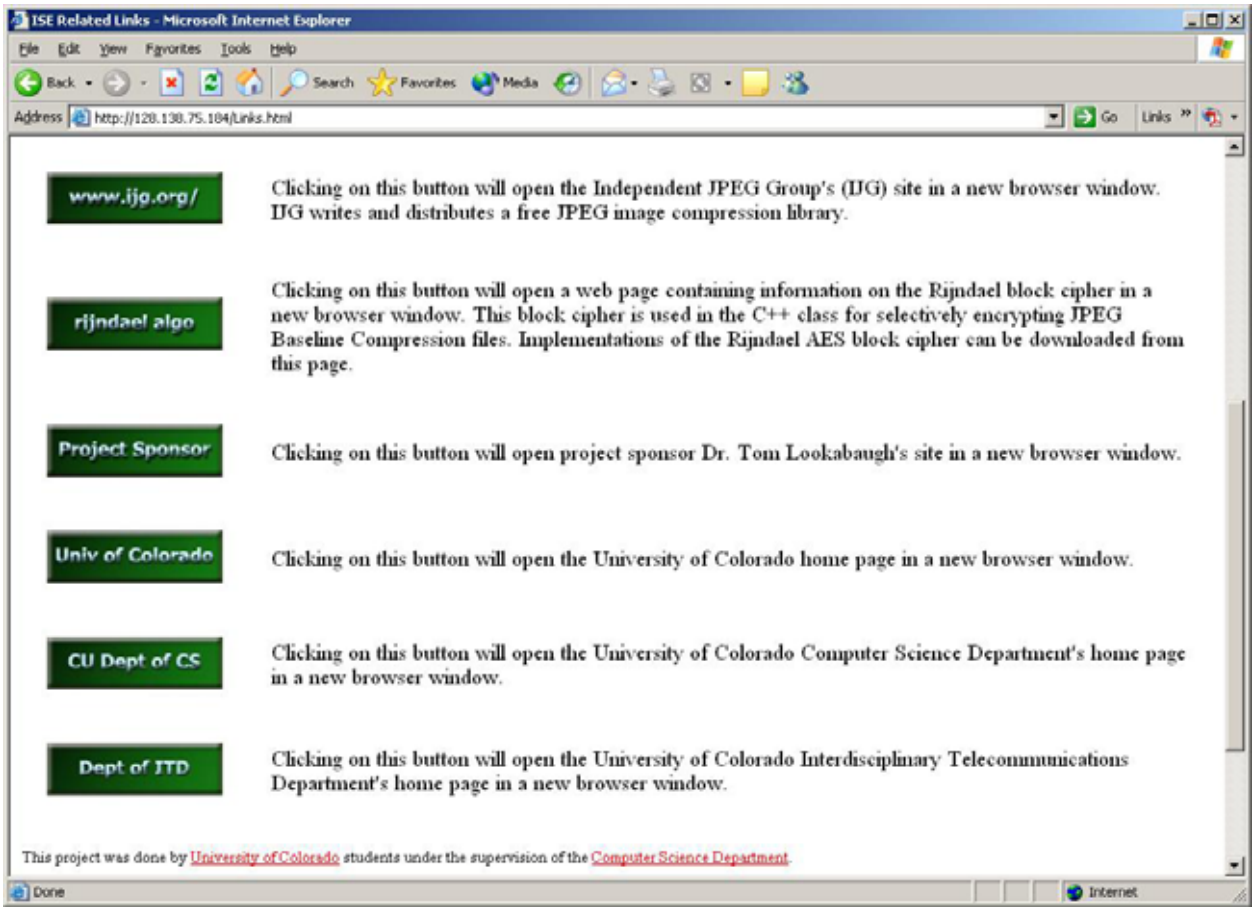


Figure 2.3.2.4: Screenshot of Links Page

The user can always return to the ISE home page, displayed in Figure 2.3.2.1, by clicking on the Home button in the menu bar. The user can access information on the Project Sponsor, Tom Lookabaugh, by clicking on the Project Sponsor button, and can read the project proposal by clicking on the Project Proposal button.

3. DESIGN OVERVIEW

This section of the document provides an overview of the design of all of Team ISE's products. This overview is given as a high-level design scheme for the ISE class production code, the JPEG Manipulator and the Team ISE web site.

3.1. High-Level Modular Decomposition

Team ISE's project breaks down into several high-level modules, each fulfilling a specific purpose for the project. The project consists of 4 main modules: the Encryptor, the Decryptor, the JPEG Manipulator test suite and the Team ISE web site. A high-level modular decomposition of Team ISE's software project is presented in the following figure:

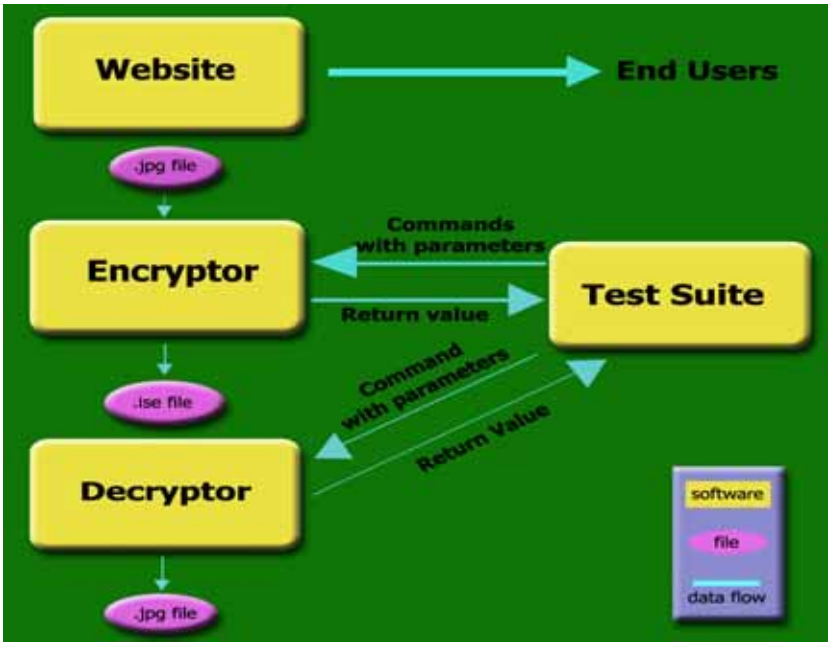


Figure 3.1: High level modular decomposition of the ISE project.

3.2. ISE Class Production Code Modules

The ISE class production code contains both the Encryptor and the Decryptor. Both of these modules are outlined below.

3.2.1. ISE Encryptor Module

The Encrpytor will be invoked as an API, as described is section 2.1.2 in this design document. The purpose of this module is to selectively encrypt a JPEG image, based upon the algorithm developed by Team ISE. The Encryptor is called by invoking one of the encrypt_file() methods found within the JPEG ISE class. The Encryptor will be included along with the Decryptor in the ISE class production code.

3.2.2. ISE Decryptor Module

The Decrpytor will be invoked as an API, as described is section 2.1.2 in this design document. The purpose of this module is to decrypt a selectively encrypted JPEG image, based upon the algorithm developed by Team ISE. The Decryptor is called by invoking

one of the decrypt_file() methods found within the JPEG ISE class. The Decryptor will be included along with the Encryptor in the ISE class production code.

3.3. JPEG Manipulator Test Suite Module

The design of the Manipulator application breaks down into six high-level sub-modules that in turn break down into a series of supporting object methods. These sub-modules are defined as:

1. Standard Windows Form Application Methods.
2. Manipulator Graphical Interface Methods.
3. Manipulator Common Methods.
4. Methods to Convert from Binary to ASCII.
5. Methods to Convert from ASCII to Binary.
6. Methods to Encrypt and Decrypt.

The following is a brief explanation of each of the Manipulator sub-modules. For a detailed description of each of the individual methods included in these sub-modules, refer to section 4 of this document.

3.3.1. Standard Windows Form Application Methods

This sub-module contains the methods necessary to support the Windows form instantiation and disposal, the main entry point of the application and any other functionality necessary to operate in the Windows environment.

3.3.2. Manipulator Graphical Interface Methods

This sub-module contains the methods necessary to support the Windows form operations and resolve events generated by Windows form components. Specifically, these methods will execute events like button clicks or provide menu option functionality.

3.3.3. Manipulator Common Methods

This sub-module contains the methods related to the core functionality of the Manipulator. These methods will be the engine for the Manipulator, implementing the low-level functionality for loading files, writing files, processing data and performing common tasks within the application.

3.3.4. Methods to Convert from Binary to ASCII

This sub-module consists of methods written to convert the byte values found within a JPEG file to ASCII characters between ‘0’and ‘F’ that represents the binary data value in hexadecimal format.

3.3.5. Methods to Convert from ASCII to Binary

This sub-module consists of methods written to convert the ASCII character value currently loaded in the Manipulator back to binary values. These functions are the reverse of the functions found in the previous section.

3.3.6. Methods to Encrypt and Decrypt

This sub-module consists of methods responsible for providing all of the encryption and decryption functionality for the Manipulator.

3.4. Team ISE Web Site Module

The web site will serve as the distributor for Team ISE's software packages, research materials and information pertaining to the final products. The site will provide links to all documentation created by the team for the software packages, the research behind implementation, sponsor information and all documents related to the ISE project.

4. DESIGN

This section of the document provides an in-depth view of design of the ISE class production code, the JPEG Manipulator and the ISE web site.

4.1. ISE Class Production Code Design

The ISE class production code will be implemented in C++ and will consist of two classes and several methods which will be outlined within this section. The ISE class is an abstract base class from which other selective encryption classes are derived. The ISE class by itself is never instantiated but is inherited by other classes and is used as an interface to define the inheriting classes. The class will implement non-file-type-specific methods and will initialize class data members.

The JPEG ISE class will inherit the ISE class and all of its non-file-type-specific methods and data members. The class will implement the selective encryption and decryption methods inherited from the ISE class, specifically designed to selectively enrypt standard baseline JPEG images.

This section details the design of the ISE class production code, including a full description of all methods and data members for both the ISE and JPEG ISE classes. The algorithm designed to selectively encrypt and decrypt JPEG files using the Huffman table information is also included in this section.

4.1.1. ISE Class Invocation

The ISE class will provide a number of different user interfaces that developers will use to employ the ISE class. The ISE class can be constructed in one of three ways:

4.1.1.1. protected ise()

Default Constructor

Pre-conditions: None.

Post-conditions:

A default ISE object is created.

Parameters: None.

Return values:

Constructor, no return type.

Description:

An ISE object can be constructed with no arguments using the default constructor on a protected level, and is not intended to be used explicitly.

4.1.1.2. ise(char * key, char * input_file_name, char * ise_file_name = NULL)

Encryption Overloaded Constructor

Pre-conditions:

The **key** must be a pointer to a character string with a maximum length of 320 characters.

Post-conditions:

An ISE object is created containing the specified data members.

Parameters:

The first argument is a pointer to the encryption key. The second argument is the name and path of the input file to be encrypted. The third argument is the ISE file name for the file generated by encryption.

Return values:

Constructor, no return type.

Description:

An ISE object may also be constructed with the data necessary to encrypt a file. This overloaded constructor only requires that the first two arguments be provided. The third argument is optional and will be set to a default value based upon the input file if it is not specified.

4.1.1.3. ise(char * key, char * ise_file_name, char * output_file_name = NULL)

Decryption Overloaded Constructor

Pre-conditions:

The **key** must be a pointer to a character string with a maximum length of 320 characters.

Post-conditions:

An ISE object is created containing the specified data members.

Parameters:

The first argument is a pointer to the encryption key. The second argument is the name and path of the ISE file to be decrypted. The third argument is the output file name for the file generated by decryption.

Return values:

Constructor, no return type.

Description:

An ISE object may also be constructed with the data necessary to decrypt an ISE file. This overloaded constructor only requires that the first two arguments be provided. The third argument is optional and will be set to a default value based upon the input file if it is not specified.

4.1.2. Public Methods of the ISE Class

There are a number of public interface exposed by the ISE class to the developer. These are the methods used to by the developer to perform the major functions of this class. These public interfaces are as follows:

4.1.2.1. virtual int encrypt_file()

Pre-conditions:

The **input_file_name** and **key** must be set using either the encryption overloaded constructor or the **set_input_file_name(char* name)** and **set_key(char* key)** functions prior to calling this method.

Post-conditions:

The **ise_file_name** file will contain the selectively encrypted file data.

Parameters: None.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The encrypt_file method will take a file and selectively encrypt the pertinent data within the file. This is a virtual method and must be implemented in the class inheriting from ISE.

4.1.2.2. virtual int encrypt_file(char * key, char * input_file_name, char * ise_file_name = NULL)

Pre-conditions: None.

Post-conditions:

An encrypted file will be created with the name and path specified by the value within the **ise_file_name** data member. If this data member is NULL, then a default file name will be created based upon the **input_file_name** data member. The **key**, **input_file_name** and **ise_file_name** data members within the class will be set to parameter values.

Parameters:

The first argument is a pointer to the encryption key. The second argument is the name and path of the input file to be encrypted. The third argument is the ISE file name for the file generated by encryption.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The encrypt_file method will take a file and selectively encrypt the pertinent data within the file. This is a virtual method and must be implemented in the class inheriting from ISE.

4.1.2.3. virtual int decrypt_file()

Pre-conditions:

The **ise_file_name** and **key** must be set using either the decryption overloaded constructor or the **set_ise_file_name(char* name)** and **set_key(char* key)** functions prior to calling this method. The **key** used in this method must be the same as the one used to encrypt the ISE file.

Post-conditions:

The **output_file_name** file will contain the selectively decrypted file data.

Parameters: None.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The decrypt method will take an instance of an ISE file and selectively decrypt the correct portion(s) of the file. This is a virtual method and must be implemented in the class inheriting from ISE.

4.1.2.4. virtual int decrypt_file(char * key, char * ise_file_name, char * output_file_name = NULL)

Pre-conditions:

The **ise_file_name** and **key** must be set using either the decryption overloaded constructor or the **set_ise_file_name(char* name)** and **set_key(char* key)** functions prior to calling this method. The **key** used in this method must be the same as the one used to encrypt the ISE file.

Post-conditions:

The **output_file_name** file will contain the selectively decrypted file data.

Parameters:

The first argument is a pointer to the encryption key. The second argument is the name and path of the ISE file to be decrypted. The third argument is the file name for the file generated by the decryption process.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The decrypt method will take an instance of an ISE file and selectively decrypt the correct portion of the file. This is a virtual method and must be implemented in the class inheriting from ISE.

4.1.2.5. int set_key(char * key)

Pre-conditions:

The **key** must point to a character string with a maximum length of 320 characters.

Post-conditions:

The **key** will be set using the new string specified. Any previous information in **key** will be lost.

Parameters:

The only argument to this method is a pointer to a character string containing the key information for either encryption or decryption.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The method will use the specified character string to create a valid key to be used by the encryption or decryption methods. This method must be called prior to calling **encrypt_file()** or **decrypt_file()** if the default constructor is used to create the ISE object**.**

4.1.2.6. int set_input_file_name(char * name)

Pre-conditions:

The **name** must be a pointer to a valid file type supported by ISE selective encryption.

Post-conditions:

The **input_file_name** will be set using the new string specified. Any previous data in **input_file_name** will be lost.

Parameters:

The only argument to this method is a pointer to a character string containing the **input_file_name,** specifying the file to be selectively encrypted.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

This method is used to set the **input_file_name**. The method must be called prior to the encryption method if the default constructor was used to create the ISE object.

4.1.2.7. int set_ise_file_name(char * name)

Pre-conditions:

The **name** must be a pointer to a valid ISE file.

Post-conditions:

The **ise_file_name** will be set using the new string specified. Any previous data in **ise_file_name** will be lost.

Parameters:

The only argument to this method is a pointer to a character string containing the **ise_file_name,** specifying the file to be selectively decrypted or the resulting selectively encrypted file.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

This method is used to set the **ise_file_name**. This method must be called prior to calling the decryption method if the default constructor was used to create the ISE object.

4.1.2.8. virtual int set_output_file_name(char * name)

Pre-conditions:

The **name** must be a pointer to a valid file type supported by ISE selective encryption.

Post-conditions:

The **output_file_name** will be set using the new string specified. Any previous **output_file_name** in the object will be lost.

Parameters:

The only argument to this method is a pointer to a character string containing the **output_file_name,** specifying the file to be created during selective decryption.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

This method is used to set the **output_file_name**. If the **output_file_name** is not specified by this method or the decrypt overloaded constructor, the program will automatically create a name based on the **ise_file_name**. The created name will be one that does not exist in the current directory. For example the string decrypted might be concatenated to the end of the **ise_file_name**. The function must be implemented in the class inheriting the ISE base class.

4.1.2.9. char * get_input_file_name()

Pre-conditions: None.
Post-conditions: None.
Parameters: None.

Return values:

The method will return the **input_file_name** character string. If the **input_file_name** is not set, the method will return an empty string. The string the returned pointer points to is owned by the class. The user need not worry about deallocating this string.

Description:

This is the accessor method for the input file name.

4.1.2.10. char * get_ise_file_name()

Pre-conditions: None.
Post-conditions: None
Parameters: None.

Return values:

The method will return the **ise_file_name** character string. If the **ise_file_name** is not set, the method will return an empty string. The string the returned pointer points to is owned by the class. The user need not worry about deallocating this string.

Description:

This is the accessor method for the **ise_file_name.**

4.1.2.11. char * get_output_file_name()

Pre-conditions: None.
Post-conditions: None.
Parameters: None.
Return values:

The method will return the **output_file_name** character string. If the **output_file_name** is not set, the method will return an empty string. The string the returned pointer points to is owned by the class. The user need not worry about deallocating this string.

Description:

This is the accessor method for the output file name.

4.1.3. Protected Methods of the ISE Class

In addition to the public interfaces, this ISE class will also contain a number of protected methods that will only be used by the classes. These methods are specific to the low-level functionality of the class and thus will not be exposed to users. These private methods are as follows:

4.1.3.1. int get_ise_file_type(char * name)

Pre-conditions:

The **name** must be a pointer to a valid ISE file.

Post-conditions: None

Parameters:

The only argument for this method is a pointer to a character string indicating the name of a valid ISE file.

Return values:

The function will return an integer indicating the type of the original file from which the specified ISE file was created.
0 will indicate an unknown or unimplemented file type.
1 will indicate a jpeg file.
The return values may be extended to accommodate other file types.

Description:

This method will return an integer corresponding to the original file type of an encrypted ISE file.

4.1.3.2. int make_ise_file_name()

Pre-conditions:

The user of the class has previously set the **input_file_name**.

Post-conditions:

The **ise_file_name** data member points to a string with a file name and file path, based upon the string pointed to by the **input_file_name**.

Parameters: None.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The file name and path created will be the same as the string pointed to by the **input_file_name** data member, except that the extension of the file will be changed to .ise. If this file already exists, then a 0 will be added on to the end of the file name, just before the extension. If this file already exists, we will keep incrementing this number and checking, until the new file name does not previously exist.

4.1.3.3. int make_output_file_name()

Pre-conditions:

The user of the class has previously set the **ise_file_name**.

Post-conditions:

The **output_file_name** data member points to a string with a file name and file path, based upon the string pointed to by the **ise_file_name**.

Parameters: None.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The file name and path created will be the same as the string pointed to by the **ise_file_name** data member, except that the extension of the file will be changed to .jpg. If this file already exists, then a 0 will be added on to the end of the file name, just before the extension. If this file already exists, we will keep incrementing this number and checking, until the new file name does not previously exist.

4.1.4. Data Members of the ISE Class

4.1.4.1. char * input_file_name

This data member defines the file to be encrypted.

4.1.4.2. char * ise_file_name

This data member defines the ISE file created after encryption.

4.1.4.3. char * output_file_name

This data member defines the file created after decryption.

4.1.4.4. char * key

This data member defines the key to be used in both encryption and decryption.

4.1.5. JPEG ISE Class Invocation

The JPEG ISE class will provide a number of different user interfaces that developers will use to employ the JPEG ISE class. The JPEG ISE class can be constructed in one of three ways:

4.1.5.1. protected jpeg_ise()

Default Constructor

Pre-conditions: None.

Post-conditions:

A default JPEG ISE object instance is created.

Parameters: None.

Return values:

Constructor, no return type.

Description:

A JPEG ISE object can be constructed with no arguments under the default constructor on a protected level, and is not intended to be used explicitly.

4.1.5.2. jpeg_ise (char * key, char * input_file_name, char * ise_file_name = NULL)

Encryption Overloaded Constructor

Pre-conditions:

The **key** must be a pointer to a character string with a maximum length of 320 characters.

Post-conditions:

A JPEG ISE object is created containing the specified data members.

Parameters:

The first argument is a pointer to the encryption key. The second argument is the name and path of the input JPEG file to be encrypted. The third argument is the ISE file name for the file generated by encryption.

Return values:

Constructor, no return type.

Description:

A JPEG ISE object may also be constructed with the data necessary to encrypt a JPEG file. This overloaded constructor only requires that the first two arguments be provided. The third argument is optional and will be set to a default value based upon the input JPEG file if it is not specified.

4.1.5.3. jpeg_ise(char * key, char * ise_file_name, char * output_file_name = NULL)

Decryption Overloaded Constructor

Pre-conditions:

The **key** must be a pointer to a character string with a maximum length of 320 characters.

Post-conditions:

A JPEG ISE object is created containing the specified data members.

Parameters:

The first argument is a pointer to the encryption key. The second argument is the name and path of the ISE file to be decrypted. The third argument is the output file name for the file generated by decryption.

Return values:

Constructor, no return type.

Description:

A JPEG ISE object may also be constructed with the data necessary to decrypt an ISE file. This overloaded constructor only requires that the first two arguments be provided. The third argument is optional and will be set to a default value based upon the input file if it is not specified.

4.1.6. Public Methods of the JPEG ISE Class

There are a number of public interface exposed by the JPEG ISE class to the developer. These are the methods used to by the developer to perform the functions of this class. Note that some of these methods will be implemented in and inherited from the ISE base class. These public interfaces are as follows:

4.1.6.1. int encrypt_file()

Pre-conditions:

The **input_file_name** and **key** must be set using either the encryption overloaded constructor or the **set_input_file_name(char* name)** and **set_key(char* key)** functions prior to calling this method.

Post-conditions:

An encrypted file will be created with the name and path specified by the value within the **ise_file_name** data member. If this data member is NULL, then a default file name will be created based upon the **input_file_name** data member.

Parameters: None.

Return values:

An integer is returned indicating a success or failure.

A zero will indicate a success.

A one will indicate a failure.

Description:

The encrypt_file method will take a standard baseline compression JPEG file and selectively encrypt the Huffman Table frames found within the file, as well as delete all application and file comment data. If the file already exists, the existing file will be overwritten. If there is not enough space, the partial file will be deleted, and an error message will be provided telling the user that there is not enough disk space. The exact algorithm used for this method is fully explained in section 4.1.9.1 of this document. A new, encrypted file will be created for this selectively encrypted JPEG image.

4.1.6.2. int encrypt_file(char * key, char * input_file_name, char * ise_file_name = NULL)

Pre-conditions: None.

Post-conditions:

An encrypted file will be created with the name and path specified by the value within the **ise_file_name** data member. If this data member is NULL, then a default file name will be created based upon the **input_file_name** data member.

Parameters:

An encrypted file will be created with the name and path specified by the value within the **ise_file_name** data member. If this data member is NULL, then a default file name will be created based upon the **input_file_name** data member. The **key**, **input_file_name** and **ise_file_name** data members within the class will be set to parameter values.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The encrypt_file method will take a standard baseline compression JPEG file and selectively encrypt the Huffman Table frames found within the file, as well as delete all application and file comment data. If the file already exists, the existing file will be overwritten. If there is not enough space, the partial file will be deleted, and an error message will be provided telling the user that there is not enough disk space. The exact algorithm used for this method is fully explained in section 4.1.9.1 of this document. A new, encrypted file will be created for this selectively encrypted JPEG image.

4.1.6.3. int decrypt_file()

Pre-conditions:

The **ise_file_name** and **key** must be set using either the decryption overloaded constructor or the **set_ise_file_name(char* name)** and **set_key(char* key)** functions prior to calling this method. The **key** used in this method must be the same as the one used to encrypt the JPEG image.

Post-conditions:

A decrypted file will be created with the name and path specified by the value within the **output_file_name** data member. If this data member is NULL, then a default file name will be created based upon the **ise_file_name** data member.

Parameters: None.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The decrypt_file method will take a standard ISE file and selectively decrypt the Huffman Table frames found within the file. The exact algorithm used for this method is fully explained in section 4.1.9.2 of this document. A new, decrypted standard JPEG image file will be created from this ISE file. If the file already exists, the existing file will be overwritten. If there is not enough space, the partial file will be deleted, and an error message will be provided telling the user that there is not enough disk space.

4.1.6.4. int decrypt_file(char * key, char * ise_file_name, char * output_file_name = NULL)

Pre-conditions: None.

Post-conditions:

A decrypted file will be created with the name and path specified by the value within the **output_file_name** data member. If this data member is NULL, then a default file name will be created based upon the **ise_file_name** data member. The **key**, **input_file_name** and **ise_file_name** data members within the class will be set to parameter values.

Parameters:

The first argument is a pointer to the encryption key. The second argument is the name and path of the ISE file to be decrypted. The third argument is the file name for the file generated by decryption the process.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

The decrypt_file method will take a standard ISE file and selectively decrypt the Huffman Table frames found within the file. The exact algorithm used for this method is fully explained in section 4.1.9.2 of this document. A new, decrypted standard JPEG image file will be created from this ISE file. If the file already exists, the existing file will be overwritten. If there is not enough space, the partial file will be deleted, and an error message will be provided telling the user that there is not enough disk space.

4.1.6.5. int set_key(char * key)

Pre-conditions:

The **key** must point to a character string with a maximum length of 320 characters.

Post-conditions:

The **key** will be set using the new string specified. Any previous information in **key** will be lost.

Parameters:

The only argument to this method is a pointer to a character string containing the key information for either encryption or decryption.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

Implemented in class ISE. The method will use the specified character string to create a valid key to be used by the encryption or decryption methods. This method must be called prior to calling **encrypt_file()** or **decrypt_file()** if the default constructor is used to create the JPEG ISE object**.**

4.1.6.6. int set_input_file_name(char * name)

Pre-conditions:

The **name** must be a pointer to a standard baseline JPEG image file.

Post-conditions:

The **input_file_name** will be set using the new string specified. Any previous data in **input_file_name** will be lost.

Parameters:

The only argument to this method is a pointer to a character string containing the **input_file_name,** specifying the JPEG file to be selectively encrypted.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

Implemented in class ISE. This method is used to set the **input_file_name**. The method must be called prior to the encryption method if the default constructor was used to create the JPEG ISE object.

4.1.6.7. int set_ise_file_name(char * name)

Pre-conditions:

The **name** must be a pointer to a valid ISE file.

Post-conditions:

The **ise_file_name** will be set using the new string specified. Any previous data in **ise_file_name** will be lost.

Parameters:

The only argument to this method is a pointer to a character string containing the **ise_file_name,** specifying the JPEG file to be selectively decrypted.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

Implemented in class ISE. This method is used to set the **ise_file_name**. This method must be called prior to calling the decryption method if the default constructor was used to create the ISE object.

4.1.6.8. int set_output_file_name(char * name)

Pre-conditions:

The **name** must be a pointer to a standard baseline JPEG file.

Post-conditions:

The **output_file_name** will be set using the new string specified. Any previous data in **output_file_name** will be lost.

Parameters:

The only argument to this method is a pointer to a character string containing the **output_file_name,** specifying the JPEG file to be created during selective decryption.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

Implemented in class ISE. This method is used to set the **output_file_name**. If the **output_file_name** is not specified by this method or the decrypt overloaded constructor, the program will automatically create a name based on the **ise_file_name**. The created name will be one that does not exist in the current directory. For example the string "decrypted" might be concatenated to the end of the **ise_file_name**.

4.1.6.9. char * get_input_file_name()

Pre-conditions: None.
Post-conditions: None.
Parameters: None.
Return values:

The method will return the **input_file_name** character string. If the **input_file_name** is not set, the method will return an empty string.

Description:

Implemented in class ISE. This is the accessor method for the input file name.

4.1.6.10. char * get_ise_file_name()

Pre-conditions: None.
Post-conditions: None
Parameters: None.
Return values:

The method will return the **ise_file_name** character string. If the **ise_file_name** is not set, the method will return an empty string.

Description:

Implemented in class ISE. This is the accessor method for the **ise_file_name.**

4.1.6.11. char * get_output_file_name()

Pre-conditions: None.
Post-conditions: None.
Parameters: None.
Return values:

The method will return the **output_file_name** character string. If the **output_file_name** is not set, the method will return an empty string.

Description:

Implemented in class ISE. This is the accessor method for the output file name.

4.1.7. Protected Methods of the JPEG ISE Class

In addition to the public interfaces, this JPEG ISE class will also contain a number of private methods. These methods are specific to the low-level functionality of the class

and thus will not be exposed to users. Some of this functionality will be implemented and inherited from the in the ISE base class. These private methods are as follows:

4.1.7.1. int get_ise_file_type(char * name)

Pre-conditions:

The **name** must be a pointer to a valid ISE file.

Post-conditions:

None

Parameters:

The only argument for this method is a pointer to a character string indicating the name of an ISE file.

Return values:

The function will return an integer indicating the type of the original file from which the specified ISE file was created. The return value will be a one to indicate a JPEG ISE file.

Description:

Implemented in class ISE. This method will return an integer corresponding to the original file type of an encrypted ISE file.

4.1.7.2. int make_ise_file_name()

Pre-conditions:

The user of the class has previously set the **input_file_name**.

Post-conditions:

The **ise_file_name** data member points to a string with a file name and file path, based upon the string pointed to by the **input_file_name**.

Parameters: None.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

Implemented in class ISE. The file name and path created will be the same as the string pointed to by the **input_file_name** data member, except that the extension of the file will be changed to .ise. If this file already exists, then a 0 will be added on to the end of the file name, just before the extension. If this file already exists, we will keep incrementing this number and checking, until the new file name does not previously exist.

4.1.7.3. int make_output_file_name()

Pre-conditions:

The user of the class has previously set the **ise_file_name**.

Post-conditions:

The **output_file_name** data member points to a string with a file name and file path, based upon the string pointed to by the **ise_file_name**.

Parameters: None.

Return values:

An integer is returned indicating a success or failure.
A zero will indicate a success.
A one will indicate a failure.

Description:

Implemented in class ISE. The file name and path created will be the same as the string pointed to by the **ise_file_name** data member, except that the extension of the file will be changed to .jpg. If this file already exists, then a 0 will be added on to the end of the file name, just before the extension. If this file already exists, we will keep incrementing this number and checking, until the new file name does not previously exist.

4.1.8. Data Members of the JPEG ISE Class

4.1.8.1. char * input_file_name

Inherited from ISE base class. This data member defines the standard baseline JPEG file to be encrypted.

4.1.8.2. char * ise_file_name

Inherited from ISE base class. This data member defines the ISE file created after encryption.

4.1.8.3. char * output_file_name

Inherited from ISE base class. This data member defines the standard baseline JPEG file to be created after decryption.

4.1.8.4. char * key

Inherited from ISE base class. This data member defines the key to be used for encryption and decryption.

4.1.9. Algorithms Developed by Team ISE Used in the ISE Class

The research conducted by Team ISE has lead to the conclusion that the Huffman tables are the best targets for selective encryption of standard baseline JPEG images. Because we do not want to increase the size of the file after encryption, Team ISE has decided, as recommended by Professor John Black, to utilize the AES (Advanced Encryption Standard) encryption method.

4.1.9.1. JPEG Selective Encryption Algorithm

1. Write a single byte of information to the output file stream to indicate the type of this ISE encrypted file. For a JPEG ISE file the byte written will be a 1. Write a second byte indicating the version number of the ISE software used.
2. Read from the input file stream one byte at a time and write the information to the output file stream until a two byte frame marker value of ffe0 through ffef, fffe, or ffc0 through ffcf (hexadecimal) is found. These JPEG markers

indicate the beginning of application data, comment data, or Huffman data respectively. If the end of file is reached, proceed to step 15.
3. If a Huffman Marker is found, proceed to step 6.
4. If an application or comment marker is found, read through the input file stream without writing the information back to the output file stream, checking for a JPEG marker (any two-byte value beginning with ff).
5. If a Huffman marker is found, proceed to step 6, otherwise, return to step 2.
6. Write the unencrypted Huffman marker to the output file stream.
7. Put the next 16 bytes of data, or plain text, into a buffer, checking each for a non-Huffman marker. A non-Huffman marker is any other two-byte, hexadecimal value below ffc0 or above ffcf.
8. If a non-Huffman marker is found, proceed to step 12.
9. If no non-Huffman marker is found, encrypt the 16-byte block using Rijndael's blockEncrypt() method to produce the cipher text.
10. Write the cipher text to the output file stream.
11. Return to step 7.
12. Encrypt the last 16-byte block containing the non-Huffman marker using Rijndael's blockEncrypt() method to produce the cipher text.
13. Write the cipher text to the output file stream.
14. Return to step 2.
15. Exit the program successfully.

4.1.9.2. JPEG Selective Decryption Algorithm

1. Read a single byte of information from the input file. The byte will be a 1 if this is a valid ISE JPEG file. Read a second byte that indicates the version number of the ISE software used to create the file. Check version number to ensure compatibility. If not compatible, cancel operation and display error message.
2. Read off of the input file stream one byte at a time and write the information to the output file stream until a Huffman marker is found or the end of file has been reached. This marker indicates the beginning of the encrypted data.
3. If the end of file is reached, exit the program.
4. If a Huffman marker is found, write the two bytes to the output file stream.
5. Put the next 16 bytes of data from the input file stream, or cipher text, into a buffer.
6. Decrypt this 16-byte block using Rijndael's blockDecrypt() method to produce the plain text.
7. Scan the plain text for a non-Huffman marker.
8. Write the plain text to the output file stream.
9. If the plain text did not contain a non-Huffman marker, return to step 5.
10. If the cipher text contains a non-Huffman marker, return to step 2.

4.2. The JPEG Manipulator Design

The purpose of the JPEG Manipulator is to provide Team ISE with a tool for testing the data manipulation of JPEG images. The ideal application will supply an easy-to-use, graphical interface that grants the user the ability to simultaneously view both the original image and manipulated image, as well as view and update the original JPEG's data. The team has chosen Microsoft's Visual C# (pronounced cee-sharp) programming language to accomplish these tasks.

Developing the Manipulator in C# will allow the use of the .NET (pronounced dot-net) framework tools, satisfying all of functionality requirements outlined for the application[1]. The .NET framework will reduce the amount of components developed by the team, since .NET offers a wide variety of functionality and tools. The Manipulator should make use of the managed code features .NET offers, to effectively reduce the cost of future program maintenance.

To implement all of the functionality requirements[1], the Manipulator requires that the team develop an extensive catalog of methods. As mentioned in section 3.3 of this document, these methods break down into six major sub-modules:

1. Standard Windows Form Application Methods.
2. Manipulator Graphical Interface Methods.
3. Manipulator Common Methods.
4. Methods to Convert from Binary to ASCII.
5. Methods to Convert from ASCII to Binary.
6. Methods to Encrypt and Decrypt.

This section of the design document outlines all of the functionality that will be created by Team ISE for the Manipulator. A complete list of the methods needed for the Manipulator is included in this section of the document. Function prototypes, pre-conditions, post-conditions, parameter descriptions, return value information and function descriptions for each of the Manipulators methods is included here.

4.2.1. Standard Windows Form Application Methods

There are a number of functions that ISE Manipulator application is required to call to begin execution of the main program. In addition, the System.Windows.Form class requires constructor and dispose methods. The Visual Studio Windows Form Designer requires a constructor method, defined as InitializeComponent(). Lastly, we will add an additional method to be invoked by the Form's constructor to initialize all of the variables used by the Manipulator.

This section of the design document defines each of the functions necessary to satisfy the requirements of a standard Windows application. Each of these function prototypes, pre-conditions, post-conditions, parameters, return values and descriptions are provided below:

[1] See Team ISE Requirements Specification for full listing of the ISE project requirements.

4.2.1.1. [STAThread] static void Main()

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.

4.2.1.2. public frmMain()

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.

4.2.1.3. private void InitializeComponent()

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 Form's constructor. It initializes all of the variables and values set with the form designer at the beginning of the program execution.

4.2.1.4. private void ISEConstructor()

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.

4.2.1.5. protected override void Dispose(bool disposing)

Pre-conditions: None.

Post-conditions:

All of the memory and resources used in the frmMain have been freed.

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.

4.2.2. ISE Manipulator Graphical Interface Methods

There are a number of functions that ISE Manipulator application is required to implement to facilitate the use of the graphical interface objects. In .NET, usually these methods are created in the Form of "events" that can occur on the parent control of any particular Windows object, as the result of a control being invoked be the user of the application. For example, when a user click's on the left mouse button over a Button control, the Button generates an interrupt event within the parent of the Button object. For these interrupts to be processed by the parent control, each desired event for any particular object that should be implemented, must be implemented within the scope of the parent control. If an event is not implemented on the parent control and it occurs during execution, this event will be ignored. All of the event methods required for the JPEG Manipulator's graphical interface are outlined in this section of the design document.

4.2.2.1. private void menuOpen_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.2. private void menuExit_Click(object sender, System.EventArgs e)

Pre-conditions:

The menuExit menu object has generated a Click event.

Post-conditions:

The application is terminated and exited successfully.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.3. private void menuAbout_Click(object sender, System.EventArgs e)

Pre-conditions:

The menuAbout menu object has generated a Click event.

Post-conditions:

The frmAbout Form has been displayed for the user to view.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.4. private void menuNewProject_Click(object sender, System.EventArgs e)

Pre-conditions:

The menuNewProject menu object has generated a Click event.

Post-conditions:

A new project file has been created by the application.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.5. private void menuOpenProject_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.6. private void menuSaveProject_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.7. private void txtChangedFile_TextChanged(object sender, System.EventArgs e)

Pre-conditions:

The txtChangedFile TextBox object has generated a TextChanged event.

Post-conditions:

A warning is displayed if the changed text reflects a file path that already exists.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function is used to resolve a TextChanged event generated by the txtChangedFile 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.

4.2.2.8. private void txtQuantizer1_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.9. private void txtQuantizer2_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.10. private void txtQuantizer3_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.11. private void txtQuantizer4_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.12. private void txtHuffman1_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.13. private void txtHuffman2_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.14. private void txtHuffman3_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.15. private void txtHuffman4_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.16. private void txtHuffman5_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.17. private void txtHuffman6_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.18. private void txtHuffman7_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.19. private void txtHuffman8_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.20. private void btnRestoreQuantizer1_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.21. private void btnRestoreQuantizer2_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.22. private void btnRestoreQuantizer3_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.23. private void btnRestoreQuantizer4_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.24. private void btnRestoreHuffman1_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.25. private void btnRestoreHuffman2_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.26. private void btnRestoreHuffman3_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.27. private void btnRestoreHuffman4_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.28. private void btnRestoreHuffman5_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.
The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.29. private void btnRestoreHuffman6_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function is used to resolve a Click event generated by the btnRestoreHuffman6 Button object. The purpose of this Button is to allow the user to restore the original data for this Huffman table to the txtHuffman6 TextBox.

4.2.2.30. private void btnRestoreHuffman7_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.31. private void btnRestoreHuffman8_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.32. private void btnUpdate_Click(object sender, System.EventArgs e)

Pre-conditions:

The btnUpdate Menu Button object has generated a Click event.

Post-conditions:

A changed picture has been updated within the application.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.33. private void btnNew_Click(object sender, System.EventArgs e)

Pre-conditions:

The btnNew Menu Button object has generated a Click event.

Post-conditions:

This function clears out all data for pictures.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.34. private void btnLoad_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.35. private void btnSave_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.36. private void btnLoadPicture_Click(object sender, System.EventArgs e)

Pre-conditions:

The btnLoadPicture Menu Button object has generated a Click event.

Post-conditions:

An image file has been loaded by the application.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.37. private void btnUpdatePicture_Click(object sender, System.EventArgs e)

Pre-conditions:

The btnUpdatePicture Menu Button object has generated a Click event.

Post-conditions:

A changed picture has been updated within the application.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function is used to resolve a Click event generated by the btnUpdatePicture Menu 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.

4.2.2.38. private void menuCut_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.39. private void menuCopy_Click(object sender, System.EventArgs e)

Pre-conditions:

The menuCopy menu object has generated a Click event.

Post-conditions:

Selected text has been copied to the system clipboard.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.2.40. private void menuPaste_Click(object sender, System.EventArgs e)

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.

Parameters:

The **sender** parameter is a pointer to the function calling this function.

The **e** parameter is for the base class to pass event data.

Return values:

Function returns void.

Description:

This function 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.

4.2.3. ISE Manipulator Common Methods

This section of the document describes the ISE Manipulator common methods. These are a collection of methods mostly called by the interface events in the Manipulator, but should be called by any method requiring any of this functionality. The prototypes and definitions of each of these methods are outlined in this section of the document.

4.2.3.1. private void LoadPicture(string OriginalFilePath, string ChangedFilePath)

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 picChanged and picChangedSmall data members. Also, all of the data contained in the original file should be loaded into the interface to display for the user.

Parameters:

The OriginalFilePath parameter is a file path of the to the image to be loaded into the picOriginal and picOriginalSmall. The ChangedFilePath parameter is a file path of the to the image to be loaded into the picChanged and picChangedSmall.

Return values:

Function returns void.

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, picChanged and picChangedSmall 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 UpdateChangedPicture() to load the picChanged picture. If a valid file doesn't exist in the ChangedFilePath parameter, then it should just load the file in the OriginalFilePath parameter. If the OriginalFilePath parameter doesn't 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.

4.2.3.2. private void UpdateChangedPicture(string FileName)

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 picChanged and the picChangedSmall data fields.

Parameters:

The FileName parameter is the name and path of a JPEG file to be loaded.

Return values:

Function returns void.

Description:

This function is used to update picChanged and picChangedSmall 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.

4.2.3.3. private bool ShowWarning(string message, string caption)

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.

Parameters:

The message parameter is explanation of the warning message.
The caption parameter is Window title of warning message box.

Return values:

Function returns True if the user has clicked Ok and False if the user has clicked Cancel.

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.

4.2.3.4. private bool ShowWarning(string message)

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.

Parameters:

The message parameter is explanation of the warning message.

Return values:

Function returns True if the user has clicked Ok and False if the user has clicked Cancel.

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.

4.2.3.5. private void ClearInterfaceData()

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.

Parameters: None.

Return values:

Function returns void.

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.

4.2.3.6. private void WriteFile(ref byte[] ByteDataToWrite)

Pre-conditions: None.

Post-conditions:

A new file with the data contained in the ByteDataToWrite array has been created.

Parameters:

The ByteDataToWrite parameter is byte array of data to be written to file.

Return values:

Function returns void.

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 txtChangedFile 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.

4.2.3.7. private void ClearData()

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.

Parameters: None.

Return values:

Function returns void.

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.

4.2.3.8. private void LoadNewProject()

Pre-conditions: None.

Post-conditions:

A previously existing SEP project file has been reloaded into the Manipulator.

Parameters: None.

Return values:

Function returns void.

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.

4.2.3.9. private void SaveNewProject()

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.

Parameters: None.

Return values:

Function returns void.

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.

4.2.3.10. private void CreateNewProject()

Pre-conditions: None.

Post-conditions:

The current project within the Manipulator is closed and a new SEP project file is created. All of the data loaded in the Manipulator should stay the same, except the txtNotes TextBox for the project notes should be cleared out for the new project file.

Parameters: None.

Return values:

Function returns void.

Description:

The purpose of this method is to allow the caller to create a new SEP project for the picture and data currently loaded in the Manipulator. If there is a project file currently open, then the user should be prompted to save before this project is closed. The txtNotes TextBox should be cleared out, as these notes belong to the last project. Then the user should be prompted to create a new project name for the new SEP project and all of the current data within the Manipulator should be saved to this new project. 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.

4.2.4. ISE Methods to Convert from Binary to ASCII

The Manipulator methods found within this section of the document are related to converting binary data to the ASCII characters displayed in the interface for the user to view. These functions, along with the methods in section 4.2.5 of this document, represent the lowest level of functionality that the application is required to perform. The prototypes and definitions of each of these methods are outlined in this section of the document.

4.2.4.1. private void SetCharValues(int OneByte, ref char HighBits, ref char LowBits)

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.

Parameters:

The OneByte parameter is an integer value between 0 and 255 (8-bits), representing the value of one byte.

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.

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.

Return values:

Function returns void.

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.

4.2.4.2. private char Convert(int Value)

Pre-conditions: None.

Post-conditions:

A character based on the hexadecimal value of the integer parameter passed in should be returned.

Parameters:

The Value parameter is an integer value between 0 and 15 (4-bits).

Return values:

Function returns a char based upon the hexadecimal value of the parameter.

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 'X' 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.

4.2.4.3. private void LoadPictureData(string FilePath)

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.

Parameters:

The FilePath parameter is the file name and path to a JPEG image.

Return values:

Function returns void.

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.

4.2.4.4. private void LoadInterfaceData(ref StringBuilder HexChars)

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.

Parameters:

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.

Return values:

Function returns void.

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[2], 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.

4.2.5. ISE Methods to Convert from ASCII to Binary

The Manipulator methods found within this section of the document are related to converting data from ASCII characters to Binary format, so that a new image can be created based upon the values currently loaded in the Manipulator's interface. These functions, along with the methods in section 4.2.4 of this document, represent the lowest level of functionality that the application is required to perform. The prototypes and definitions of each of these methods are outlined in this section of the document.

[2] For full information about the JPEG standard, refer to the "JPEG Still Image Data Compression Standard" book referenced in the related readings in section 8 of this document.

4.2.5.1. private byte SetByteValue(char HighBits, char LowBits)

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.

Parameters:

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.

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.

Return values:

Function returns a byte value based upon the parameters passed in.

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.

4.2.5.2. private int Convert(char Hex)

Pre-conditions: None.

Post-conditions:

An integer representing the binary value of the hexadecimal ASCII character parameter passed will be returned.

Parameters:

The Hex parameter is an ASCII character between 0 and F.

Return values:

Function returns an int based upon the hexadecimal value of the char parameter.

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 –1 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.

4.2.5.3. private bool CreateChangedPicture(ref byte [] File)

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.

Parameters:

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 Manipulator's interface.

Return values:

Function returns void.

Description:

The purpose of this method is to take all of the data currently loaded in the Manipulator's 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.

4.2.5.4. private void CreateISEImage()

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 txtChangedFile TextBox field.

Parameters: None.

Return values:

Function returns void.

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 CreateChangedPicture() 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 UpdateChangedPicture() 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.

4.2.6. Data Members of the JPEG Manipulator

This section of the design document defines all data members the JPEG Manipulator application will use to perform its functions. Many of these data members are Windows form components used for the graphical user interface, along with a smattering of primitive data types.

The main form, frmMain, of the Manipulator application is inherited from the System.Windows.Forms class. It is a Windows form control that contains all of the following data members, along with all of the previously described methods.

4.2.6.1. public class frmMain: System.Windows.Forms.Form

This class contained within the JPEG Manipulator namespace is the definition of the main form of the Manipulator application. This form contains all of the data members for the application. Through this form, the entire JPEG Manipulator application is executed. This class is broken down into all of the data members and methods found in section 4.2 of this document.

4.2.6.2. Menu Controls

The Manipulator will employ a menu control to fulfill its functionality requirement[3] of behaving like a standard Windows application. The declaration of this data member is as follows:

```
private System.Windows.Forms.MainMenu menuFrmMain;
```

Aside from the Menu control, there are a number of MenuItem controls connected to this Menu. All of the declarations for these menu items are as follows:

```
private System.Windows.Forms.MenuItem menuFile;
private System.Windows.Forms.MenuItem menuOpen;
private System.Windows.Forms.MenuItem menuExit;
private System.Windows.Forms.MenuItem menuUpdate;
private System.Windows.Forms.MenuItem menuOpenProject;
private System.Windows.Forms.MenuItem menuSaveProject;
private System.Windows.Forms.MenuItem menuNewProject;
private System.Windows.Forms.MenuItem menuHelpMain;
private System.Windows.Forms.MenuItem menuAbout;
private System.Windows.Forms.MenuItem menuHelp;
private System.Windows.Forms.MenuItem menuEdit;
private System.Windows.Forms.MenuItem menuCut;
private System.Windows.Forms.MenuItem menuCopy;
private System.Windows.Forms.MenuItem menuPaste;
```

4.2.6.2. Picture Box Controls

To serve the purpose of displaying images within the Manipulator, a series of Picture Box controls will be employed for image viewing. The declaration of these data members is as follows:

```
private System.Windows.Forms.PictureBox picOriginal;
```

[3] For a full listing of all Team ISE product requirements, please see the Final Requirements Specification referenced in the Related Readings in section 8 of this document.

```
private System.Windows.Forms.PictureBox picChanged;
private System.Windows.Forms.PictureBox picOriginalSmall;
private System.Windows.Forms.PictureBox picChangedSmall;
```

4.2.6.3. Save/Open File Dialog Controls

For the purpose of browsing for files to open or save, the Manipulator will employ standard dialog box controls. The declaration of these data members is as follows:

```
private System.Windows.Forms.SaveFileDialog saveFileDialog1;
private System.Windows.Forms.OpenFileDialog openFileDialog1;
private System.Windows.Forms.OpenFileDialog openFileDialog;
```

4.2.6.4. Component Control

In order to be properly classified as a valid Windows Form derived from the Form class, each Windows form requires the IContainer to be one of its data members. This allows the Form to be used as a component. The declaration of this data member is as follows:

```
private System.ComponentModel.IContainer components;
```

4.2.6.5. ToolTip Control

To further ease the use of the Manipulator, all controls viewable to the user in the interface will have some tool tip information as part of their data, so that users will know what purpose all of the controls in the application serve. The declaration of this data member is as follows:

```
private System.Windows.Forms.ToolTip toolTips;
```

4.2.6.6. Tab Controls

To break up all of the information and Windows controls that will be displayed for the Manipulator user, the application is designed to use a series of tab controls that will hold the various categories of data found within a JPEG image. Certain tabs, namely the tabProject, tabEncrypt and tabDecrypt will not contain data directly from the image, but the data stored here will relate to the specific image. The main tab control used for the Console, Original Picture and Manipulated Picture tab pages will be declared on the frmMain, which is declared as follows:

```
private System.Windows.Forms.TabControl tabMain;
```

In addition, the Console tab will have another set of tab controls that contain all of the various JPEG data presentation controls, and will be declared as follows:

```
private System.Windows.Forms.TabControl tabSubConsole;
```

All of the tab pages will be placed on one of these previous two tab controls. All of the tab pages that will be placed on tabMain will be declared as follows:

```
private System.Windows.Forms.TabPage tabConsol;
private System.Windows.Forms.TabPage tabOriginal;
private System.Windows.Forms.TabPage tabChanged;
```

All of the TabPage controls that will be placed on tabSubConsole will be declared as follows:

```
private System.Windows.Forms.TabPage tabProject;
private System.Windows.Forms.TabPage tabFile;
private System.Windows.Forms.TabPage tabQuantizer;
private System.Windows.Forms.TabPage tabEncodedData;
private System.Windows.Forms.TabPage tabHuffman1;
private System.Windows.Forms.TabPage tabHuffman2;
private System.Windows.Forms.TabPage tabApplicationData;
private System.Windows.Forms.TabPage tabMisc;
private System.Windows.Forms.TabPage tabEncrypt;
private System.Windows.Forms.TabPage tabDecrypt;
```

4.2.6.7. TextBox Controls

This application will use a large number of TextBox controls to store all of the potential data contained within a JPEG image. The declarations for all of these TextBox controls used within the Manipulator application are outlined in this section of the document. The TextBox controls that will be found on the tabFile TabPage control will be declared as:

```
private System.Windows.Forms.TextBox txtChangedFile;
private System.Windows.Forms.TextBox txtOriginalFile;
private System.Windows.Forms.TextBox txtComments;
private System.Windows.Forms.TextBox txtFileSize;
```

The TextBox controls that will be found on the tabHuffman1 and tabHuffman2 TabPage controls will be declared as:

```
private System.Windows.Forms.TextBox txtHuffman1;
private System.Windows.Forms.TextBox txtHuffman2;
private System.Windows.Forms.TextBox txtHuffman3;
private System.Windows.Forms.TextBox txtHuffman4;
private System.Windows.Forms.TextBox txtHuffman5;
private System.Windows.Forms.TextBox txtHuffman6;
private System.Windows.Forms.TextBox txtHuffman7;
private System.Windows.Forms.TextBox txtHuffman8;

private System.Windows.Forms.TextBox txtHuffmanOriginal1;
```

```
private System.Windows.Forms.TextBox txtHuffmanOriginal2;
private System.Windows.Forms.TextBox txtHuffmanOriginal3;
private System.Windows.Forms.TextBox txtHuffmanOriginal4;
private System.Windows.Forms.TextBox txtHuffmanOriginal5;
private System.Windows.Forms.TextBox txtHuffmanOriginal6;
private System.Windows.Forms.TextBox txtHuffmanOriginal7;
private System.Windows.Forms.TextBox txtHuffmanOriginal8;
```

The TextBox controls that will be found on the tabEncodedData TabPage control will be declared as:

```
private System.Windows.Forms.TextBox txtEncodedData;
private System.Windows.Forms.TextBox txtScanHeader;
private System.Windows.Forms.TextBox txtOriginalEncodedData;
private System.Windows.Forms.TextBox txtOriginalHeader;
```

The TextBox controls that will be found on the tabApplicationData TabPage control will be declared as:

```
private System.Windows.Forms.TextBox txtApplicationData1;
private System.Windows.Forms.TextBox txtApplicationData2;
private System.Windows.Forms.TextBox txtApplicationData3;
private System.Windows.Forms.TextBox txtApplicationData4;
private System.Windows.Forms.TextBox txtApplicationData5;
private System.Windows.Forms.TextBox txtApplicationData6;
private System.Windows.Forms.TextBox txtApplicationData7;
private System.Windows.Forms.TextBox txtApplicationData8;
private System.Windows.Forms.TextBox txtApplicationData9;
private System.Windows.Forms.TextBox txtApplicationData10;
```

The TextBox controls that will be found on the tabQuantizer TabPage control will be declared as:

```
private System.Windows.Forms.TextBox txtQuantizer1;
private System.Windows.Forms.TextBox txtQuantizer2;
private System.Windows.Forms.TextBox txtQuantizer3;
private System.Windows.Forms.TextBox txtQuantizer4;

private System.Windows.Forms.TextBox txtQuantizerOriginal1;
private System.Windows.Forms.TextBox txtQuantizerOriginal2;
private System.Windows.Forms.TextBox txtQuantizerOriginal3;
private System.Windows.Forms.TextBox txtQuantizerOriginal4;
```

The TextBox controls that will be found on the tabMisc TabPage control will be declared as:

```
private System.Windows.Forms.TextBox txtNumberLines;
private System.Windows.Forms.TextBox txtRestartMod8;
private System.Windows.Forms.TextBox txtHierarchial;
private System.Windows.Forms.TextBox txtExpand;
private System.Windows.Forms.TextBox txtRestart;
private System.Windows.Forms.TextBox txtError;
```

The TextBox controls that will be found on the tabProject TabPage control will be declared as:

```
private System.Windows.Forms.TextBox txtProjectPath;
private System.Windows.Forms.TextBox txtNotes;
```

4.2.6.8. Label Controls

To allow the user to understand what all of the Windows controls used within the Manipulator's interface do, a Label should be made for most of the TextBox controls, along with a few other controls. The declaration for each of these Label controls is outline in this section of the document. The Label controls that will be found on the tabFile TabPage control will be declared as:

```
private System.Windows.Forms.Label lblOriginalFile;
private System.Windows.Forms.Label lblChangedFile;
private System.Windows.Forms.Label lblFileSize;
private System.Windows.Forms.Label lblComments;
```

The Label controls that will be found on the tabProject TabPage control will be declared as:

```
private System.Windows.Forms.Label lblFilePath;
private System.Windows.Forms.Label lblNotes;
```

The Label controls that will be found on the tabHuffman1 and tabHuffman2 TabPage controls will be declared as:

```
private System.Windows.Forms.Label lblHuffman1;
private System.Windows.Forms.Label lblHuffman2;
private System.Windows.Forms.Label lblHuffman3;
private System.Windows.Forms.Label lblHuffman4;
private System.Windows.Forms.Label lblHuffman5;
private System.Windows.Forms.Label lblHuffman6;
private System.Windows.Forms.Label lblHuffman7;
private System.Windows.Forms.Label lblHuffman8;

private System.Windows.Forms.Label lblHuffmanMarker1;
private System.Windows.Forms.Label lblHuffmanMarker2;
private System.Windows.Forms.Label lblHuffmanMarker3;
```

```
private System.Windows.Forms.Label lblHuffmanMarker4;
private System.Windows.Forms.Label lblHuffmanMarker5;
private System.Windows.Forms.Label lblHuffmanMarker6;
private System.Windows.Forms.Label lblHuffmanMarker7;
private System.Windows.Forms.Label lblHuffmanMarker8;

private System.Windows.Forms.Label lblHuffmanOriginalMarker1;
private System.Windows.Forms.Label lblHuffmanOriginalMarker2;
private System.Windows.Forms.Label lblHuffmanOriginalMarker3;
private System.Windows.Forms.Label lblHuffmanOriginalMarker4;
private System.Windows.Forms.Label lblHuffmanOriginalMarker5;
private System.Windows.Forms.Label lblHuffmanOriginalMarker6;
private System.Windows.Forms.Label lblHuffmanOriginalMarker7;
private System.Windows.Forms.Label lblHuffmanOriginalMarker8;

private System.Windows.Forms.Label lblHuffmanOriginal1;
private System.Windows.Forms.Label lblHuffmanOriginal2;
private System.Windows.Forms.Label lblHuffmanOriginal3;
private System.Windows.Forms.Label lblHuffmanOriginal4;
private System.Windows.Forms.Label lblHuffmanOriginal5;
private System.Windows.Forms.Label lblHuffmanOriginal6;
private System.Windows.Forms.Label lblHuffmanOriginal7;
private System.Windows.Forms.Label lblHuffmanOriginal8;
```

The Label controls that will be found on the tabEncodedData TabPage control will be declared as:

```
private System.Windows.Forms.Label lblScanHeader;
private System.Windows.Forms.Label lblEncodedData;
private System.Windows.Forms.Label lblOriginalHeader;
private System.Windows.Forms.Label lblOriginalEncodedData;
```

The Label controls that will be found on the tabApplicationData TabPage control will be declared as:

```
private System.Windows.Forms.Label lblApplicationData1;
private System.Windows.Forms.Label lblApplicationData2;
private System.Windows.Forms.Label lblApplicationData3;
private System.Windows.Forms.Label lblApplicationData4;
private System.Windows.Forms.Label lblApplicationData5;
private System.Windows.Forms.Label lblApplicationData6;
private System.Windows.Forms.Label lblApplicationData7;
private System.Windows.Forms.Label lblApplicationData8;
private System.Windows.Forms.Label lblApplicationData9;
private System.Windows.Forms.Label lblApplicationData10;
```

```
private System.Windows.Forms.Label lblApplicationMarker1;
private System.Windows.Forms.Label lblApplicationMarker2;
private System.Windows.Forms.Label lblApplicationMarker3;
private System.Windows.Forms.Label lblApplicationMarker4;
private System.Windows.Forms.Label lblApplicationMarker5;
private System.Windows.Forms.Label lblApplicationMarker6;
private System.Windows.Forms.Label lblApplicationMarker7;
private System.Windows.Forms.Label lblApplicationMarker8;
private System.Windows.Forms.Label lblApplicationMarker9;
private System.Windows.Forms.Label lblApplicationMarker10;
```

The Label controls that will be found on the tabQuantizer TabPage control will be declared as:

```
private System.Windows.Forms.Label lblQuantizer1;
private System.Windows.Forms.Label lblQuantizer2;
private System.Windows.Forms.Label lblQuantizer3;
private System.Windows.Forms.Label lblQuantizer4;

private System.Windows.Forms.Label lblQuantizerOriginal1;
private System.Windows.Forms.Label lblQuantizerOriginal2;
private System.Windows.Forms.Label lblQuantizerOriginal3;
private System.Windows.Forms.Label lblQuantizerOriginal4;

private System.Windows.Forms.Label lblQuantizerOriginalMarker1;
private System.Windows.Forms.Label lblQuantizerOriginalMarker2;
private System.Windows.Forms.Label lblQuantizerOriginalMarker3;
private System.Windows.Forms.Label lblQuantizerOriginalMarker4;

private System.Windows.Forms.Label lblQuantizerMarker1;
private System.Windows.Forms.Label lblQuantizerMarker2;
private System.Windows.Forms.Label lblQuantizerMarker3;
private System.Windows.Forms.Label lblQuantizerMarker4;
```

The Label controls that will be found on the tabMisc TabPage control will be declared as:

```
private System.Windows.Forms.Label lblNumberLines;
private System.Windows.Forms.Label lblRestartMarker;
private System.Windows.Forms.Label lblRestart;
private System.Windows.Forms.Label lblNumberLinesMarker;
private System.Windows.Forms.Label lblError;
private System.Windows.Forms.Label lblRestartMod8;
private System.Windows.Forms.Label lblHierarchialMarker;
private System.Windows.Forms.Label lblHierarchial;
private System.Windows.Forms.Label lblExpandMarker;
```

```
private System.Windows.Forms.Label lblExpand;
```

4.2.6.9. Button Controls

The Manipulator interface will also provide a series of standard Windows Button controls for the user to click on to begin execution of certain functionality. All of the declarations for these Button controls are outlined in this section of the document. The Button controls that will be found on the tabQuantizer TabPage control will be declared as:

```
private System.Windows.Forms.Button btnRestoreQuantizer1;
private System.Windows.Forms.Button btnRestoreQuantizer2;
private System.Windows.Forms.Button btnRestoreQuantizer3;
private System.Windows.Forms.Button btnRestoreQuantizer4;
```

The Button controls that will be found on the tabHuffman1 and the tabHuffman2 TabPage controls will be declared as:

```
private System.Windows.Forms.Button btnRestoreHuffman1;
private System.Windows.Forms.Button btnRestoreHuffman2;
private System.Windows.Forms.Button btnRestoreHuffman3;
private System.Windows.Forms.Button btnRestoreHuffman4;
private System.Windows.Forms.Button btnRestoreHuffman5;
private System.Windows.Forms.Button btnRestoreHuffman6;
private System.Windows.Forms.Button btnRestoreHuffman7;
private System.Windows.Forms.Button btnRestoreHuffman8;
```

The Button controls that will be found on the tabProject TabPage control will be declared as:

```
private System.Windows.Forms.Button btnLoad;
private System.Windows.Forms.Button btnNew;
private System.Windows.Forms.Button btnSave;
private System.Windows.Forms.Button btnLoadPicture;
private System.Windows.Forms.Button btnSavePicture;
private System.Windows.Forms.Button btnUpdatePicture;
```

4.2.6.10. Additional Forms

In addition to the frmMain form, the Manipulator will have a couple of other small forms for providing a help Window, an about Window, and a loading form for the user to view. All of the declarations for the additional forms contained within the Manipulator are as follows:

```
private System.Windows.Forms.Form MainAbout;
private System.Windows.Forms.Form MainHelp;
private System.Windows.Forms.Form frmLoad;
```

4.2.6.11. Image Type Members

To store the images that will be loaded into each of the PictureBox controls, the Manipulator will contain four data members to store each of these image's data. The declarations of each of these members are as follows:

```
private System.Drawing.Image JPEG;
private System.Drawing.Image ISE;
private System.Drawing.Image JPEGsmall;
private System.Drawing.Image ISEsmall;
```

4.2.6.12. Miscellaneous Members

Finally, a large portion of the data members mentioned within this document and/or already contained within the Manipulator application will be a slew of miscellaneous data members to store any additional data needed for the application. Since they do not fall under any other particular sections, the declarations of all of these data members are outlined here. The data member declarations are as follows:

```
private Queue FileOrder;

private string ChangedFileName;

private StringBuilder OriginalDataStream;

private int NumberOfLines;
private int RestartInterval;
private int FileSize;
private int ExpandImage;
private int RestartMod8;

private const int MAX_HUFFMAN = 8;
private const int MAX_QUANTIZER = 4;
private const int MAX_APPDATA = 10;

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

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


private int FrameSize;
private int Count;
private int Value;
private int High;
```

```
private int Low;

private FileStream OriginalFile;
private FileStream NewFile;

private const int MAX_FILE_SIZE = 10485760;          // 10 meg
private const int AVE_FILE_SIZE = 5242880;   // 5 meg

private byte [] NewData;

private bool LoadingInterface;
private bool LoadingProject;
```

These sections sum up all of the data members needed to complete the JPEG Manipulator application. All of the data members required for this application are listed here. The final product should be implemented with little-to-no difference from the data members listed above.

4.3. Team ISE Web Site Design

To support the required functionality of the ISE web site, there are a series of web pages that need to be implemented. The web site will be implemented in HTML 4.01 Transitional to ease any future maintenance required by the sponsor. The following is a description of the design of the Team ISE web site.

4.3.1. The ISE Web Site Index Page

The ISE index page, located at http://128.138.75.184, will be the default start page of the ISE web site. To conform to various web server standards, this index will be named index.html. This page contains an introduction to the website.

4.3.2. The ISE Menu Bar

The ISE menu bar was generated using Xara Menu maker. The menu bar's top level consists of links to the other web pages. The "Documents", "Downloads", and "Links" buttons contain submenus which allow the user to directly connect to respective documents, downloads, and links without visiting the actual pages. The various buttons in the menu are:

1. The "Home" button links to index.html page.
2. The "Project Proposal" button will open the final project proposal document, in PDF form.
3. The "Documentation" button will display the DocumentIndex.html.
4. The "Project Sponsor" button will display the Sponsor.html.
5. The "Team Info" button will display the Team_ISE.html page.
6. The "Downloads" button will display the Download.html page.
7. The "Links" button will display the Links.html page.
8. The "Contact button will display the Contact.html page.

4.3.3. The ISE Project Proposal Document

The project proposal document will be shown in PDF format by clicking on the "Project Proposal" button on the menu. This will cause the document to be displayed in a new browser window. This document will be named ProjectProposal.pdf and will not contain links to other places within the ISE web site.

4.3.4. The ISE Documentation Page

The ISE documentation page will contain all of the final documents created by the team during the course of this project. This page will have several links contained within it. However, at the time of creating this document the team cannot be sure about the final number of links that will be created for this page. The names of each of the buttons on this page will correspond to the documents that they link to. This page will be named DocumentIndex.html.

4.3.5. The ISE Project Sponsor Page

The project sponsor page will provide a short description of the sponsor, Tom Lookabaugh, the work he is currently involved in and a link to his web page. Information on this page can be displayed as either images or text. This page will be named Sponsor.html.

4.3.6. The Team ISE Info Page

The team information page will provide a short description of the Team ISE members, a picture of the team and links to various web pages. Information on this page can be displayed as either images or text. This page will be named Team_ISE.html.

4.3.7. The ISE Download Page

The ISE download page will contain the final production code and Manipulator application installer, along with a few other minor items, such as the .NET framework that is required before installing the Manipulator. This page will have several links contained within it. However, at the time of creating this document the team cannot be sure about the final number of links that will be created for this page. In addition to the download items, the page will contain some screenshots and product information. The names of each of the buttons on this page will correspond to the particular action chosen by the user. Information on this page can be displayed as either images or text. This page will be named Download.html.

4.3.8. The ISE Links Page

The ISE links page will contain links to various related web pages. This page will have several links contained within it. However, at the time of creating this document the team cannot be sure about the final number of links that will be created for this page. Each link will conform to the button links on the menu page and will open in a new instance of the browser if the link redirects the user to a different web site. This page will be named Links.html.

5. FILE DESCRIPTIONS

Since both the ISE production code and the JPEG Manipulator will be using input files and producing output files, we have dedicated this section of the document to defining how these files should be represented. These files break down into four categories: JPEG Standard Image Files, JPEG ISE Encrypted Files, Test Suite Manipulated Images and Test Suite Project Files.

5.1. JPEG Standard Image Files

Both the Encryptor and the Manipulator will require standard JPEG image files as input and the Decryptor should produce standard JPEG images as output. A valid JPEG file, as defined within this document, is one that conforms to the ISO JPEG Baseline Still Image Compression Standard[4]. Both the Encryptor and Manipulator require standard JPEG images but do not discriminate against file names or extensions. Behavior and output of the Encryptor and the Manipulator, when processing files that do not conform to the ISO JPEG standard, will be considered undefined.

5.2. JPEG ISE Encrypted Files

The Encryptor and the Manipulator will produce selectively encrypted JPEG image files and both the Decryptor and the Manipulator will take selectively encrypted JPEG files as input for processing. A valid JPEG ISE encrypted file should maintain the structure exactly as its corresponding decrypted JPEG image file with three exceptions. First, there should be a one-byte file descriptor prefixed on the original image file, which describes the type of ISE file being processed. Second, every data frame within the file should be processed in accordance to the algorithm outlined in section 4.1.9 of this document. Third, the file extension should be changed to .ise.

5.3. Test Suite Manipulated Images

When using the Manipulator to alter images by changing data within the different frames of the file, the Manipulator will produce images based upon the structure of the file currently loaded. Although most of the time these files will conform to the format of a standard JPEG image, the Manipulator should not restrict the user from creating the file in anyway desired. The user should be allowed to input any data desired and the Manipulator should not discriminate against any modifications in any frame of the original image, as long as the data is an ASCII character between 0 and F. Please note that this means that files created by the Manipulator in testing mode may or may not be able to be loaded with a standard JPEG image viewer. Of course, files processed in decryption mode by the Manipulator should conform to the JPEG standard image files outlined in section 5.1 of this document.

5.4. Test Suite Project Files

Since the Manipulator will supply the user with the ability to input comments about a particular project and save all of the currently loaded project information, the Manipulator will create files other than images or encrypted files. The Manipulator will also create

[4] For full information about the JPEG standard, refer to the "JPEG Still Image Data Compression Standard" book referenced in the related readings in section 8 of this document.

project files, with the extension .sep, that will contain all of the project data for future use. The user will have the ability to create these files from the Manipulator's save project option.

Two considerations will determine the format of these files: The exact format of the JPEG file and whether or not the fields have been manipulated. This second condition is necessary to avoid writing redundant data and to minimize the size of the .sep file. The format of the file will be an ASCII file with the following data:

1. Project notes data followed by a new line.
2. Original JPEG file name and path followed by a new line.
3. The manipulated JPEG file name and path followed by a new line.
4. For the quantizer tables, write the number that have been modified followed by a new line.
5. If the number is greater than zero, write the quantizer table(s) number followed by a new line.
6. Write the modified table values to the file, followed by a new line.
7. For the Huffman tables, write the number that have been modified followed by a new line.
8. If the number is greater than zero, write the Huffman table number followed by a new line.
9. Write the modified table values to the file, followed by a new line.
10. If the encoded data has been modified, write the modified data to the file, followed by a new line.

6. SUMMARY

This project design document has outlined the necessary functionality required to complete the ISE class production code, the JPEG Mainpulator test suite and the Team ISE web site. The user interface for each of these modules, found in section 2, has been provided as a design guideline for the ISE final products. The high-level modular decomposition, found in section 3, gives a general overview of the high-level design for each of the different modules in the project. The design section, located in section 4, provides an in-depth, low-level description of each module to supply a guideline for the upcoming development process. In addition, section 5 defines how input and output files should be formatted and the standards those files should conform to. In writing the document, the team has tried to be as specific as possible about the design of each of the project modules. Team ISE's hope is that this design document will provide enough information to successfully complete the project with little-to-no future change in the design of any module.

7. GLOSSARY

AES

AES is an abbreviation for Advanced Encryption Standard. AES is an encryption system that utilizes block ciphering. http://csrc.nist.gov/CryptoToolkit/aes/

ANSI C++

ANSI is an abbreviation for the American National Standards Institute. C++, pronounced "cee-plus-plus," is a programming language that was created Bjarne Stroupstrup at AT&T Bell Laboratories in 1983**.** ANSI C++ is the current standard C++ programming language as defined by the American National Standards Institute.

Baseline Compression

The Baseline Compression of a JPEG image is a subset of the sequential compression algorithm as it is define by the ISO standard for JPEG images.

C#

C#, pronounced "cee-sharp," is a programming language that was created by Microsoft Incorporated in 1999. C# is an object-oriented programming language that enables programmers to quickly build a wide range of applications based on the latest Microsoft .NET technologies.

Cipher Text

A cipher text is the resulting data of some plain text data that has undergone a process of encryption.

Compression

A technique designed to reduce the amount of memory needed to store data. Typically, these algorithms utilize patterns in a file to reduce the size.

Cryptography

The practice and study of encoding data such that the original data can only be decoded by trusted individuals, for the purpose of data secrecy.

Cryptosystem

A Cryptosystem is a system for encrypting and decrypting data.

Decryption

A procedure used in cryptography to convert cipher text into plain text.

Encryption

A procedure used in cryptography to convert plain text into cipher text.

GUI

GUI is an abbreviation for Graphical User Interface.

IJG

IJG is an abbreviation for the Independent JPEG Group. IJG is an informal group that writes and distributes a widely used free C++ library that provides JPEG image compression utilities. http://www.ijg.org/

ISO

ISO is an abbreviation for the International Organization for Standardization. ISO is the world's largest developer of standards, particularly the development of technical standards.

JPEG

JPEG is an abbreviation for the Joint Photographic Experts Group. A JPEG image is an image that has undergone a compression technique designed specifically to compress image file data.

Key

A key is some data known only to trusted individuals.

Military Secrecy

A level of secrecy where all of the original data is hidden.

MP3

MP3 is an abbreviation for the MPEG-1 Audio Layer-3. MP3 is a standard for compressing raw audio data.

MPEG

MPEG is an abbreviation for the Moving Picture Experts Group. MPEG is a standard for compressing raw digital video and audio data.

Plain Text

Plain text data is the original, unencrypted data.

Rijndael

The Rijndael, pronounced "rain doll," is the original name of a type of block cipher encryption system. Rijndael is now known as the AES encryption system. http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

Post-condition

In reference to a method or function, the post-condition is a condition that the system should be in, if the function of method executes properly.

Pre-condition

In reference to a method or function, the pre-condition is a condition that has occurred before this method or function is called.

Selective Encryption

A cryptosystem which employs cunning methods to encrypt small, yet vital portions of data to reduce the amount of data encrypted while still rendering the file useless. Selective Encryption typically uses the knowledge of a file format and specifically how the data contained in the file relates to the files purpose and uses this knowledge to decide which portion of the file is encrypted.

Visual Studio .NET

Visual Studio .NET is Microsoft Corporation's latest integrated development environment for creating a wide variety of different types of software applications in multiple programming languages.

ZIP

ZIP is a standard file format for compressing a file without discriminating against the original file's type.

8. RELATED READINGS

[Chang and Li 96]

Chang, H. and Li, X. *On the Application of Image Decomposition to Image Compression and Encryption*. 1996.

Describes image degradation based on compression and encryption.

[Chang and Li 2000]

Chang, H. and Li, X. *Partial Encryption of Compressed Images and Videos*. 2000.

Describes a partial encryption scheme used on compressed multimedia files.

[Droogenbroek and Benedett 2002]

Droogenbroek, M. and Benedett, R. *Techniques for Selective Encryption of Uncompressed and Compressed Images*. 2002.

[Kailasanathan and Naini 2003]

Kailasanathan, C. and Naini, R. *Compression Performance of JPEG Encryption Scheme*. 2003.

Describes compression performance of JPEG encryption.

[Daigaku and Griffith and Jarchow and Kadhim and Pouzeshi]

Daigaku, S., Griffith, G., Jarchow, J., Kadhim, J. and Pouzeshi A. *Requirement Specification.* 2003.

Describes the requirement for Team ISE and for the ISE project.

[Daigaku and Griffith and Jarchow and Kadhim and Pouzeshi]

Daigaku, S., Griffith, G., Jarchow, J., Kadhim, J. and Pouzeshi A. *System Architecture.* 2003.

Describes the high-level system architecture for the ISE project.

[Li and Knipe and Cheng 97]

Li, X., Knipe, J. and Cheng, H. *Image Compression and Encryption Using Tree Structures*. 1997.

Describes compression methods that utilize tree structures.

[Lookabaugh and Sicker and Keaton and Guoand and Vedula 2003]
Lookabaugh, T., Sicker, D., Keaton, D., Guoand, W. and Vedula, I. *Security Analysis of Selectively Encrypted MPEG-e Streams*. 2003.

Description of the methods and results of applying selective encryption to MPEG-2 streams.

[Miano 99]
Miano, J. *Compressed Image File Formats*. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

Provides a description of the JPEG file format.

[Norcen and Uhl 2003]
Norcen, R. and Uhl, A. *Selective Encryption of the JPEG2000 Bitstream*. 2003.

Describes a selective encryption scheme on JPEG2000 files.

[Pennebaker and Mitchell 93]
Pennebaker, W. and Mitchell J. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, New York, 1993,

Provides a thorough description of the JPEG file format and its components.

[Podesser and Schmidt and Uhl 2002]
Podesser, M., Schmidt, H. and Uhl, A. *Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments*. 2002.

Describes Bitplane Encryption.

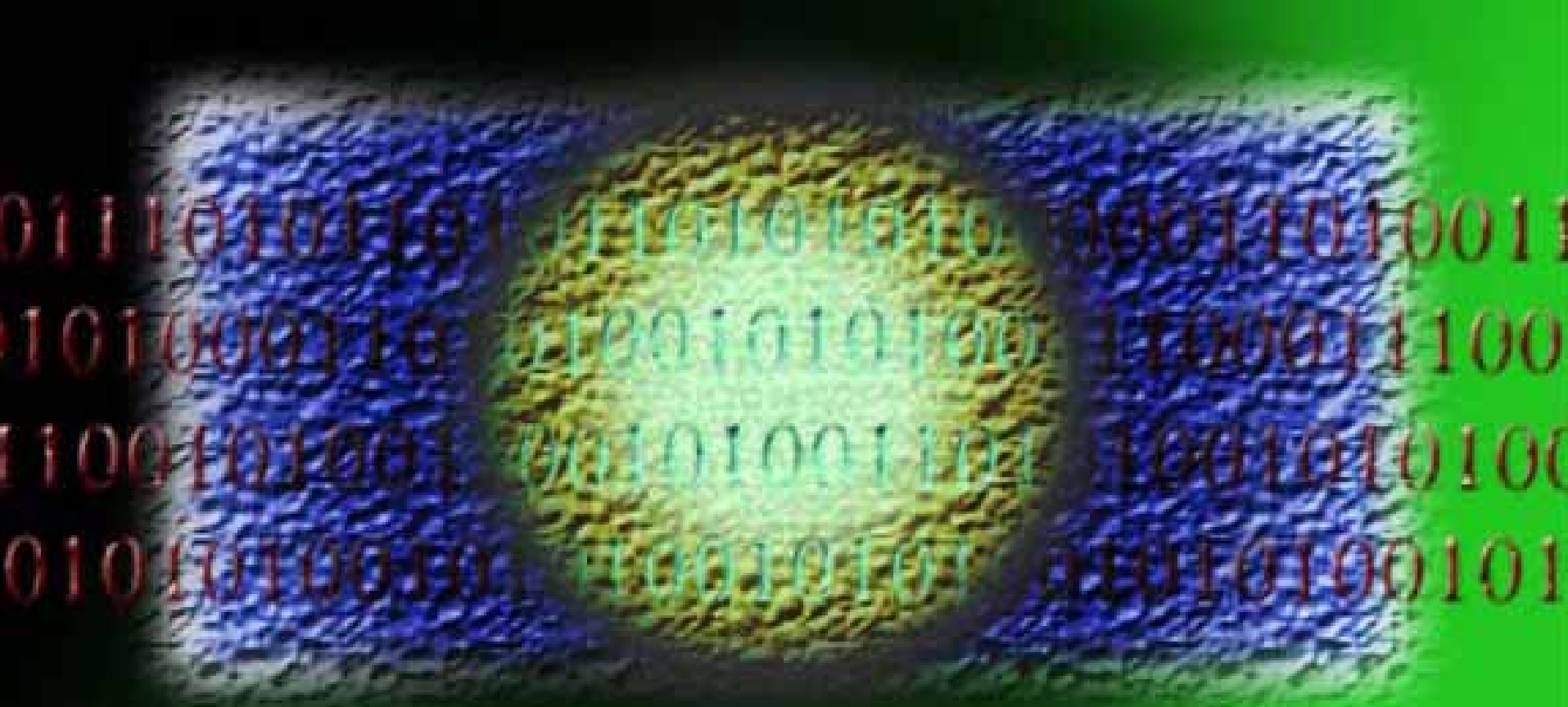
[Seo and Kim and Yoo and Dey and Agrawal 2003]
Seo, Y., Kim, D., Yoo, J., Dey, S., Agrawal, A. *Wavelet Domain Imag Encryption by Subband Selection and Data Bit Selection*. 2003.

Describes Wavelet Domain and Data Bit encryption methods.

Design Presentation

Team ISE

Image Selective Encryption



Team ISE

Image Selective Encryption

Team ISE

Image Selective Encryption

Joe Jarchow

Joseph Kadhim

Geoffrey Griffith

Shinya Daigaku

Andrew Pouzeshi

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

Problem:

- Multimedia files are very large
- Encryption is expensive
  - Processing time
  - File size
- No widely accepted solutions
  - Encrypt entire file
  - No encryption

Affected User Scenarios:

- **Images on websites**
- **File sharing**
- **Cable TV**

Solution:

- **Selective Encryption**

Definition from MPEG paper:

Selective encryption applies encryption to a subset of a file with the expectation that the entire file will be rendered useless to anyone who cannot decrypt that subset.

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

Selective Encryption Requirements:

- **Perceivable degradation of file**
- **Encryption of less than 10 %**
- **Minimize required computation**
- **Minimize increase in file size**
- **Cryptanalytic approach**

Encryption of Compressed File Types:

- Independent of time (JPEG)
  - Must affect image related target
  - Can use a block or stream cipher
- Synchronous (MPEG)
  - Target could affect the image
  - Target could affect time components
  - Might require stream cipher

Structure of Compressed File Types:

- **Published international standards**
- **Partitioned into standard components**
  - ***Descriptive***
  - ***Mathematical***

JPEG Standard:

Standard implementation of JPEG compression

http://www.ijg.org

JPEG Structure:

- **Markers, headers and data**
- **Example:**

```
ff e0
00 10
4a 46 49 46 00 01 01 01 00 48 00 48 00 00
```

Marker:

- **Indicates which component**
- **Example marker:**

ff e0 (indicates Application Data)

Header:

- **Indicates size of parameters to follow**
- **Example header:**

00 10 -- (16 bytes of data will follow)

Data:

- **The information itself**
- **Example data:**

4a 46 49 46 00 01 01 01 00 48 00 48 00 00

(16 bytes of information indicating what application created the file.)

Encrypting *During* Compression:

- **Would not produce standard file**
- **Requires reimplementation**

Encrypting *After* Compression:

- **Layered approach**
- **Creates intermediate file**
  - **Allows different extension**
- **Algorithm can be easily reviewed**
- **Applicable to non-synchronous files**

General Development Approach:

- **Study Compression Standard**
- **Study earlier approaches**
- **Create a testing toolkit**
- **Evaluate each target:**
  - **Percentage of file**
  - **Perceivable damage**
- **Design selective encryption algorithm**
- **Cryptanalytic approach**

Cryptanalytic Approach:

- **White hat**
- **Black hat**
- **Review by crypto community**
- **Correction of algorithm**

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- **Target Selection Process**
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

Criteria For Bad Targets:

- **Optional markers**
- **Not used in Baseline JPEG images**
- **Does not affect visibility of the image**
- **Easily guessed or forged by a hacker**

Determining Initial Bad Targets:

- Resources:
  - JPEG Still Image Data Compression Standard
  - Compressed Image File Formats
  - ISO DIS 80918-1 Requirements and Guidelines
  - ISO DIS 80918-2 Compliance Testing
  - http://www.funducode.com/freec/fileformats/format3/format3b.htm

- APP - Application
  - No affect to visibility
- COM - Comments
  - No affect to visibility
- DAC - Define Arithmetic Conditioning Tables
  - Not part of Baseline Compression
- DHP - Define Hierarchical Progression
  - Not part of Baseline Compression
- DNL - Define Number of Lines
  - Easily forged (set size)

- DRI - Define Restart Interval
  - Easily forged (set size)
- EOI - End of Image
  - Easily forged (always last marker)
- EXP - Expand
  - Not part of Baseline Compression
- JPG - Reserved for Future Extensions
  - Not used in Baseline Compression

- **RES - Reserved**
  - **Not used in Baseline Compression**
- **RST - Restart**
  - **Not part of Baseline Compression**
- **TEM - Temporary**
  - **Not used in Baseline Compression**
- **SOS - Start of Scan**
  - **Easily reconstructed**
- **Markers themselves are predictable**

Remaining Targets for Selective Encryption:

- **Encoded Data Stream**
- **Quantizer Tables**
- **Huffman Tables**

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- **JPEG Statistical Analysis**
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

JPEG Target Statistical Analysis:

- **Target Analysis Toolkit**
  - **Convert**
  - **Analyze**
  - **Manipulator**

Convert:

- **C++ program**
- **Convert Binary to Hexadecimal**
- **File information for a single JPEG image**

This is an ASCII representation (in hexadecimal) of the binary values found in the file : Dust.jpg

Markers Found:===============

ff d8 -- Start of Image

ff e0 -- Application Data -- 00 10 -- (16 bytes) -- 4a 46 49 46 00 01 01 01 00 48 00 48 00 00

ff db -- Define Quantization Table -- 00 43 -- (67 bytes) -- 00 06 04 05 06 05 04 06 06 05 06 07 07 06 08 0a 10 0a 0a 09 09 0a 14 0e 0f 0c 10 17 14 18 18 17 14 16 16 1a 1d 25 1f 1a 1b 23 1c 16 16 20 2c 20 23 26 27 29 2a 29 19 1f 2d 30 2d 28 30 25 28 29 28

ff db -- Define Quantization Table -- 00 43 -- (67 bytes) -- 01 07 07 07 0a 08 0a 13 0a 0a 13 28 1a 16 1a 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28

ff c0 -- Huffman Table -- Baseline DCT -- 00 11 -- (17 bytes) -- 08 01 cb 02 4a 03 01 22 00 02 11 01 03 11 01

ff c4 -- Huffman Table -- 00 1f -- (31 bytes) -- 00 00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0a 0b

Statistical Analysis

Joseph K

Analyzer:

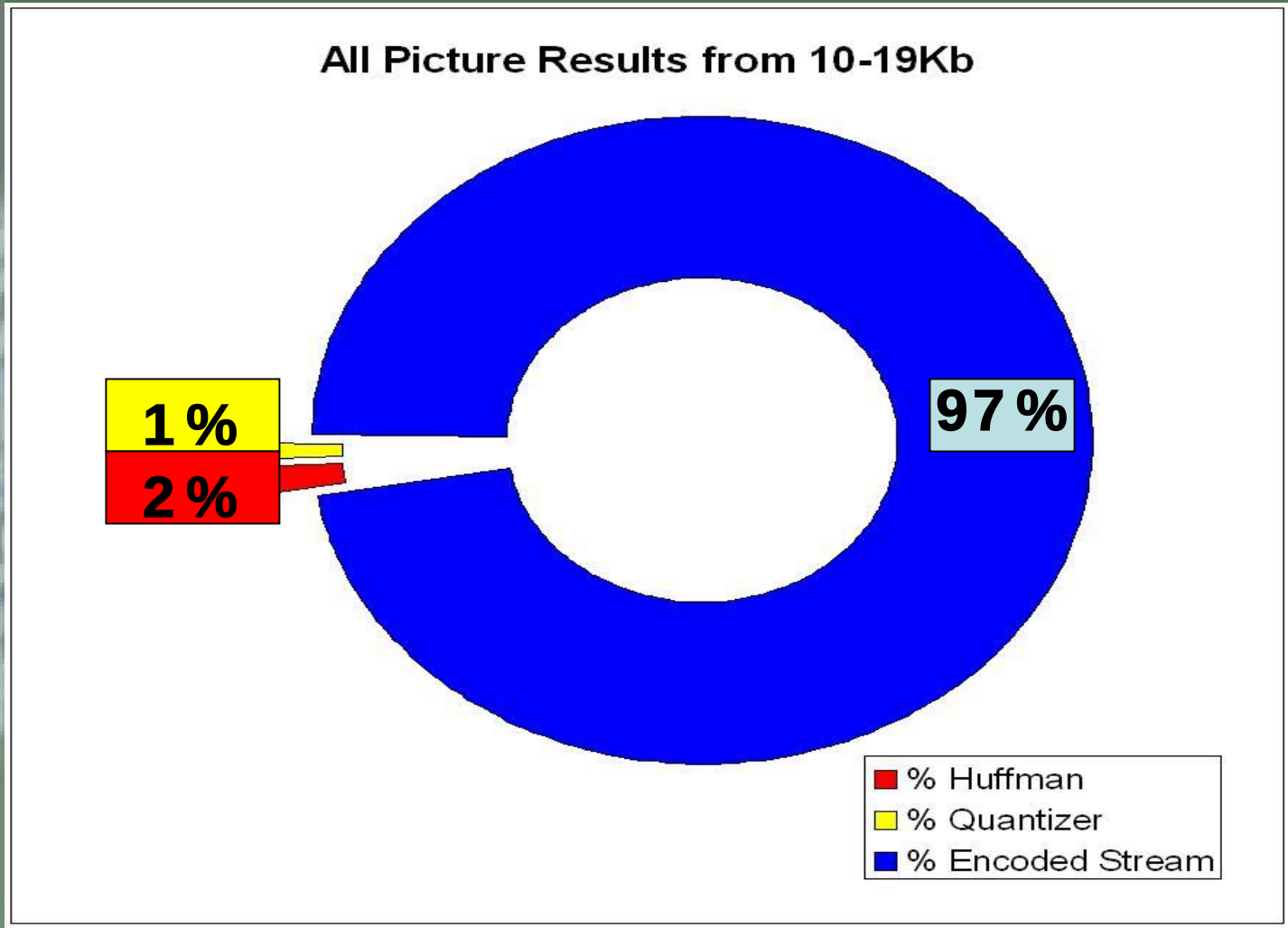
- **File information for multiple JPEG's**
  - **Average file size**
  - **Average number of Huffman tables**
  - **Average size of Huffman tables**
  - **Average number of Quantizer tables**
  - **Average size of Quantizer tables**
  - **Average size of the encoded stream**
  - **Average number of markers**
  - **Number of files processed**

Analyzer (cont):

- **Percent of the file dedicated to:**
  - **Huffman tables**
  - **Quantizer tables**
  - **Encoded Stream**

Test Cases for JPEG Analysis:

- **Over 2500 JPEG images selected**
  - **Internet web sites**
  - **Digital photographs**
  - **Manmade images**
- **Size ranges:**
  - **10-19KB, 100 KB, 1 MB, and larger**
- **Resolution Ranges:**
  - **320x240, 640x480, and 800x640 pixels**



Statistical Analysis

Joseph K

Encoded Data Stream:

- **SOI (Start of Image) marker**
- **Compressed data stream**
- **Takes up a large portion of the file**
- **Averaged 90% of the file!**

Quantizer Tables:

- **DQT (Define Quantization Table) markers**
- **Defines Resolution**
  - **Luminance**
  - **Chrominance**
- **Averaged 0.88% of the file**
- **Unpredictable affects on image**
- **Might not visually damage the image!**
- **Can be replaced with another Quantizer!**

Huffman Tables:

- **DHT (Define Huffman Table) markers**
- **Used to encode/decode the image data**
- **Averaged 1.84% of the file**
- **Considerable damage to image**
- **Mathematically derived from the image**
- **This makes the Huffman Tables a perfect target for Selective Encryption**

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- **JPEG Manipulator Design**
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

Requirements:

- **Testing tool**
- **Graphical user interface**
- **Displays each component**
- **Easy manipulation of JPEG files**
- **See changes side by side**

Modules:

- **Standard Windows methods**
- **Graphical User Interface**
- **Common methods**
- **Convert binary to ASCII**
- **Convert ASCII to binary**
- **Encrypt and Decrypt methods**

Standard Windows Methods:

- **Required functions like main()**
- **Initialization functions**
- **Constructors and Destructors**

Graphical User Interface:

- **Methods called during user interaction**
- **Event handlers**
  - **menus**
  - **buttons**
  - **text boxes**

Common Methods:

- **Create/Load/Save**
  - **project(s)**
  - **picture(s)**
- **Show warning(s)**
- **Clear interface data**
- **Updated manipulated picture**

Convert Binary to ASCII:

Convert ASCII to binary

- **Methods to load images to interface**
- **Create images from interface**

Encrypt and Decrypt methods

- **Calls production code methods**

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- **JPEG Manipulator Demonstration**
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

JPEG Selective Encryption:

- **Remove application data**
- **Remove comment data**
- **Leave initial Huffman marker**
- **Encrypt:**
  - **Huffman data (except initial marker)**
  - **Next non-Huffman marker and header**

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- **Encryption Algorithm Selection**
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

Requirements:

- **Secure**
- **No increase in file size**
- **Recommendation from Prof. John Black**

AES (Rijndael):

- **NIST selection of AES standard**
- **Block Cipher**
- **Rijmen and Daemen**
- **Open source optimized implementation**
- **Variable block length (128, 192, 256)**
- **Only whole byte operations**

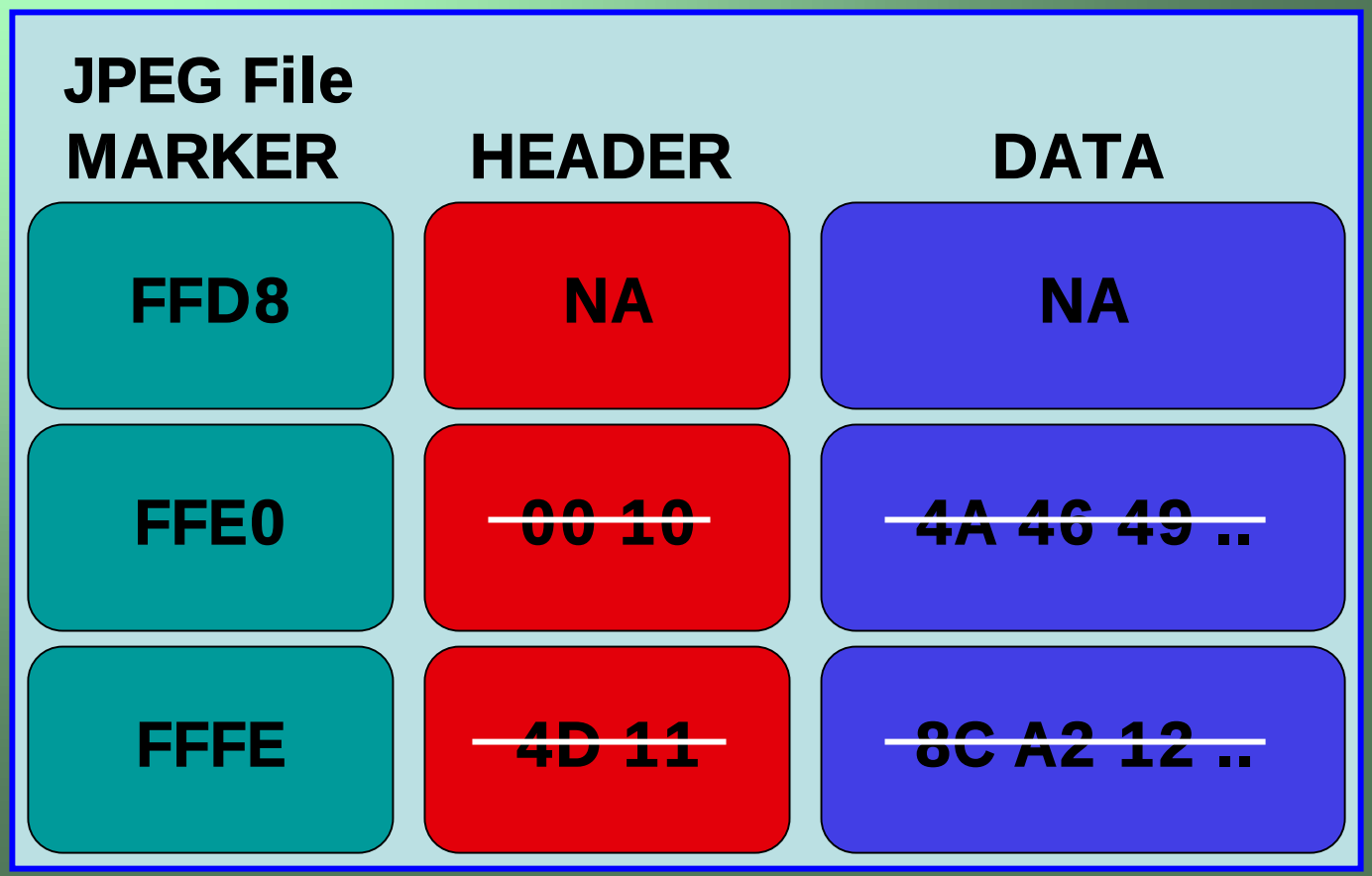
Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

Encryption Algorithm:

- **Write file-type-byte to “.ise” file**
  - **‘1’ for JPEG**
- **Read from input file**
- **Write unencrypted to “.ise” file**

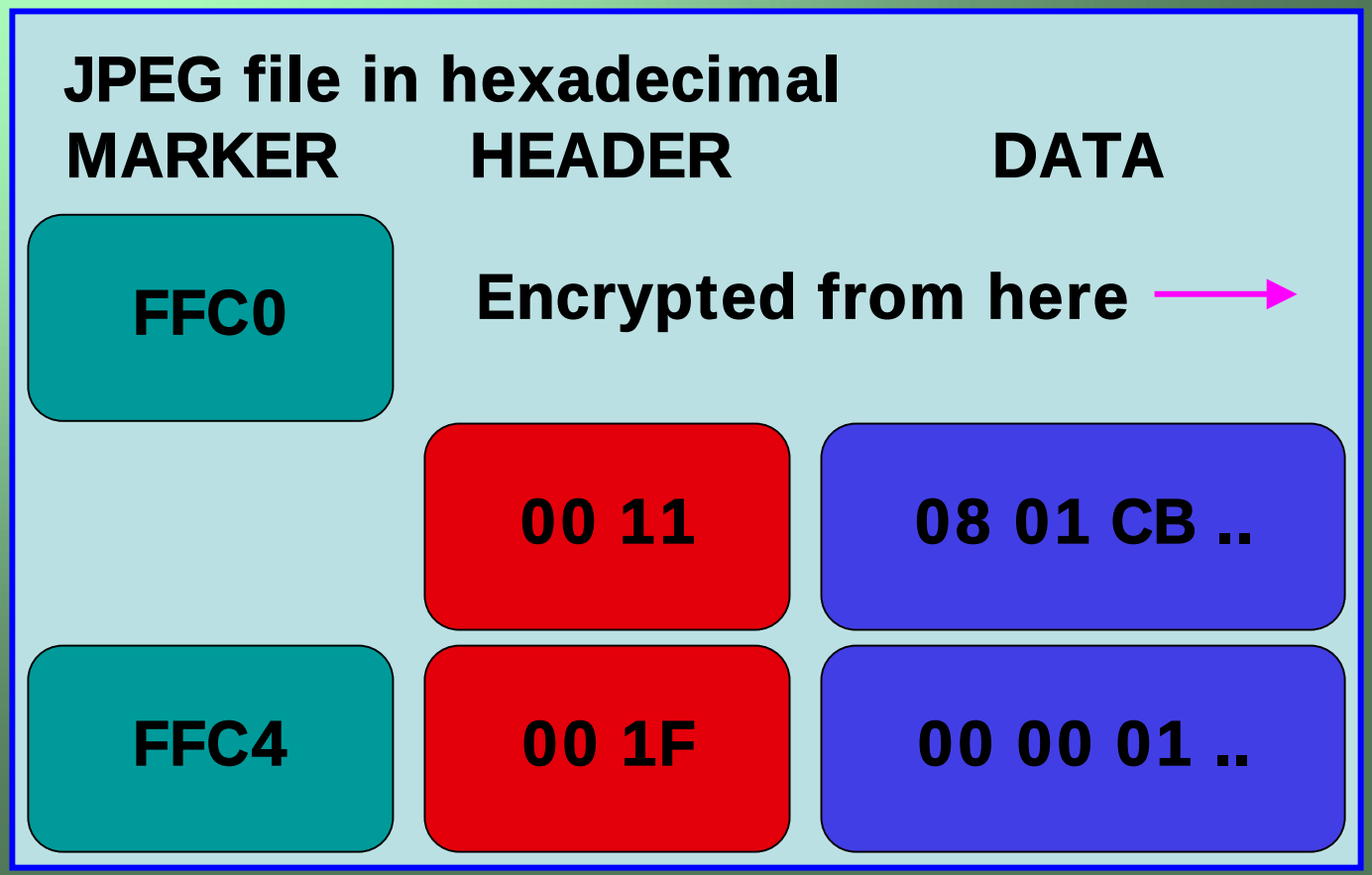
Remove App and Comment Data:

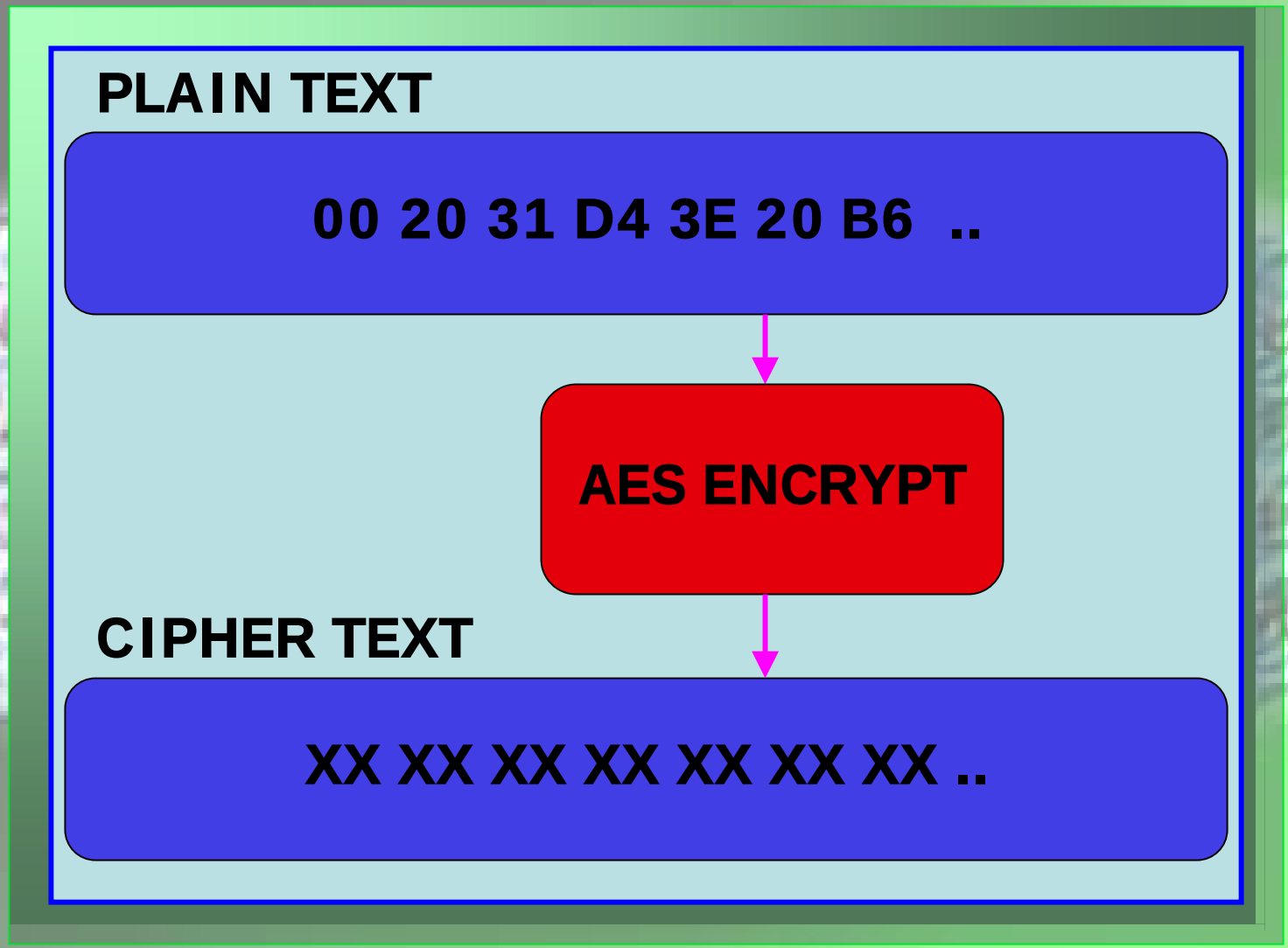


Encryption Algorithm (cont):

- **Read/Write until marker [ffc0 - ffcf]**
  - **Indicates Huffman specification**
    - **ffc0 -- baseline frame**
    - **ffc4 -- Huffman table**

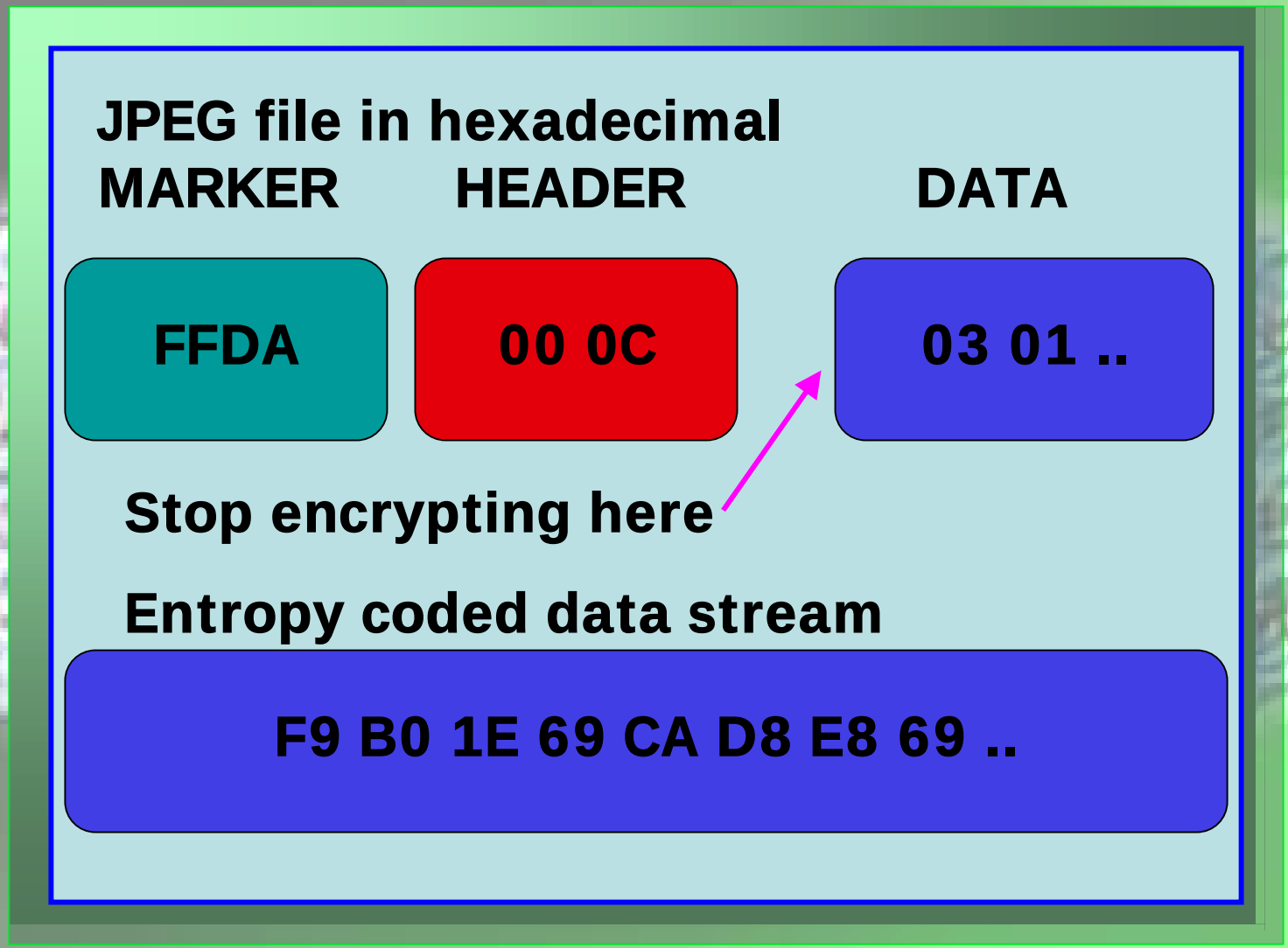
Start Encrypting After FFC0:




PLAIN TEXT
00 20 31 D4 3E 20 B6 ..
AES ENCRYPT
CIPHER TEXT
XX XX XX XX XX XX XX ..

Encryption Algorithm (cont):

- **Write until non-Huffman marker**
  - **Below ffc0**
  - **Above ffcf**

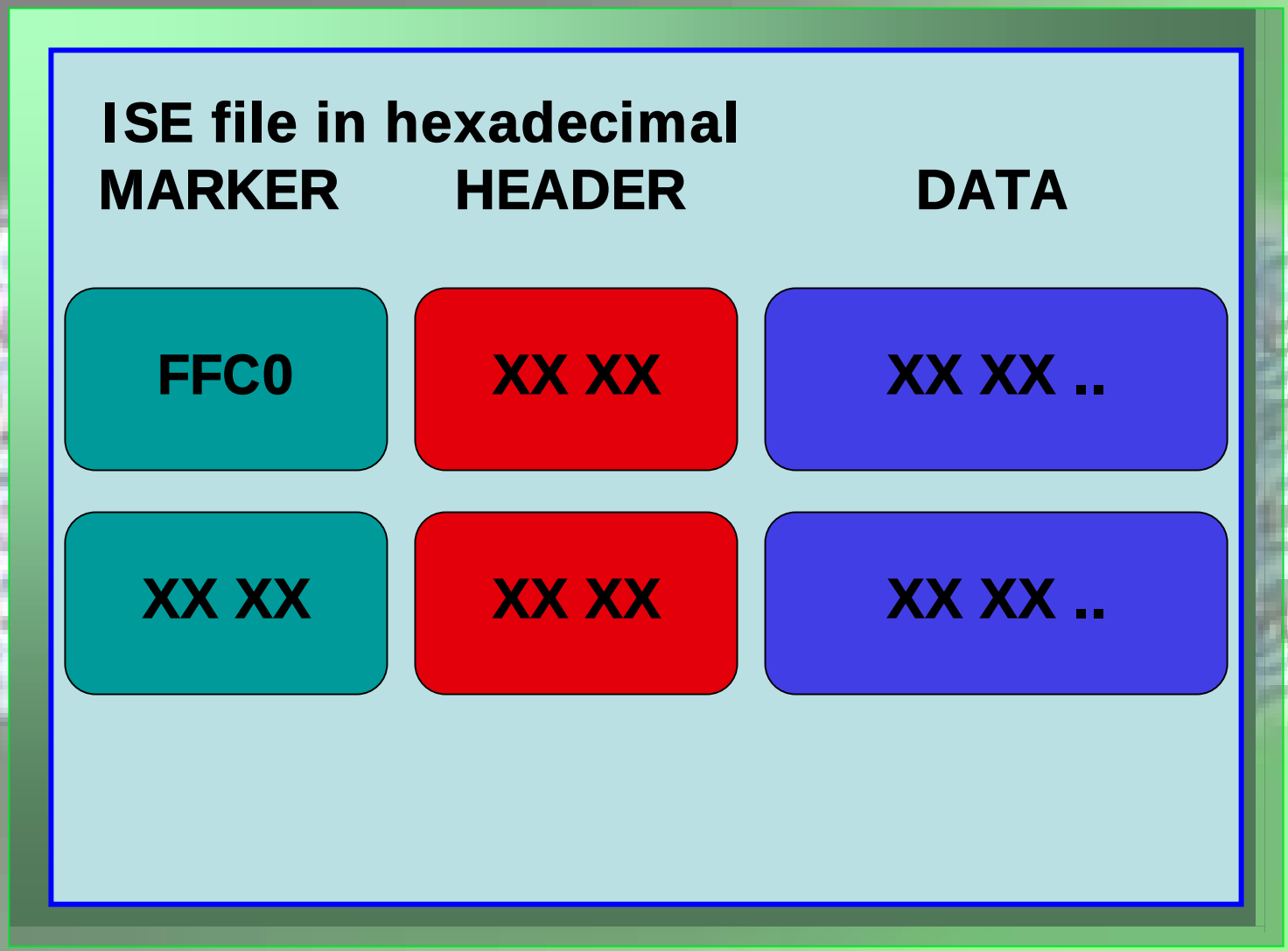

JPEG file in hexadecimal
MARKER
HEADER
DATA
FFDA
00 0C
03 01 ..
Stop encrypting here
Entropy coded data stream
F9 B0 1E 69 CA D8 E8 69 ..

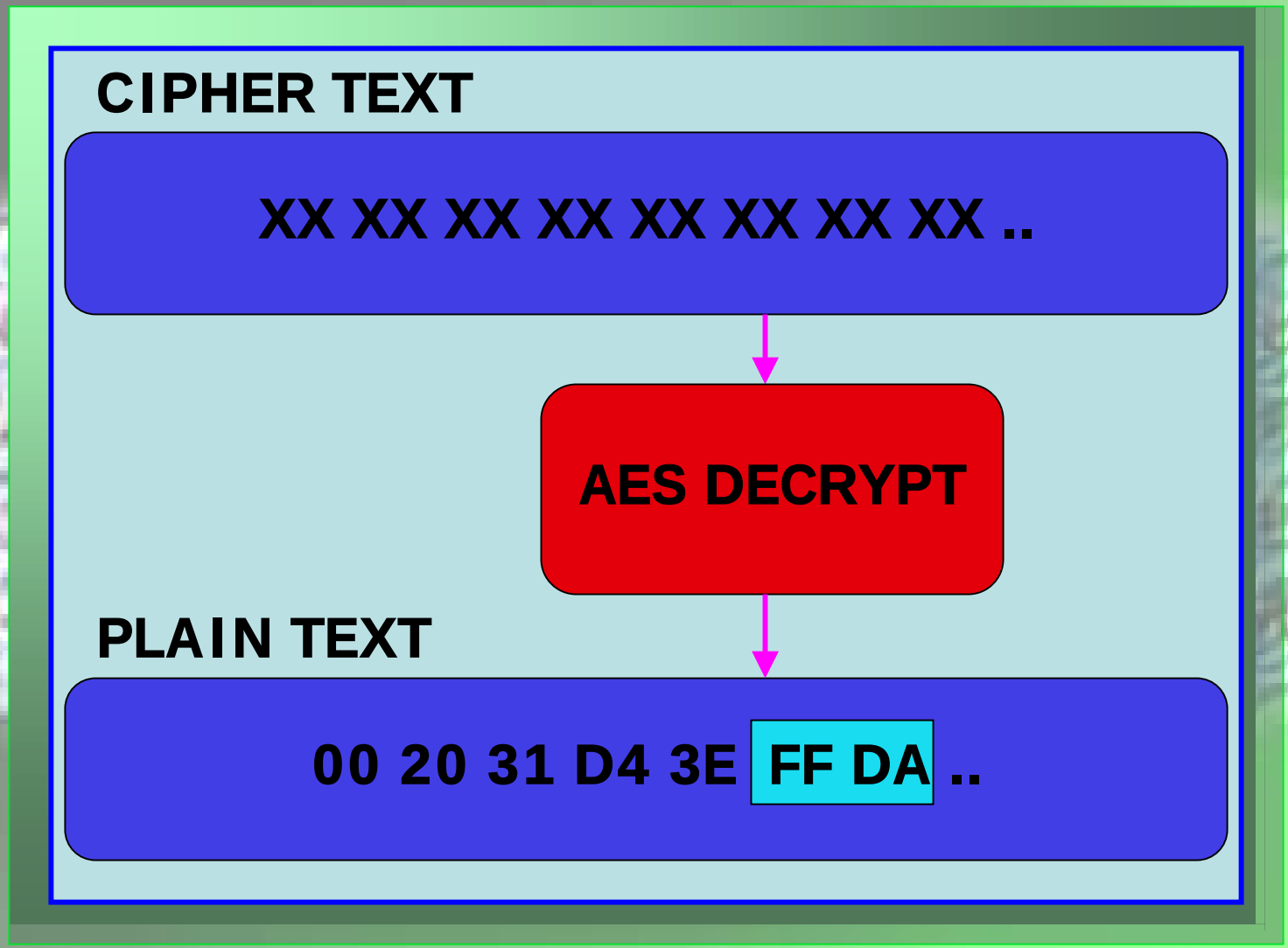
Encryption Algorithm (cont):

- **Read/Write unencrypted**
  - **Until end of file (ffd9)**
  - **Unless another Huffman marker**
- **Efficiency**
  - **97% evaluated by only a few if statements**

Decryption Algorithm:

- **Read file-type-byte from “.ise” file**
  - **‘1’ for JPEG**
- **Read/Write until marker [ffc0 - ffcf]**
  - **Indicates start of encrypted data**


ISE file in hexadecimal
MARKER
HEADER
DATA
FFC0
XX XX
XX XX ..
XX XX
XX XX
XX XX ..


CIPHER TEXT
XX XX XX XX XX XX XX XX ..
AES DECRYPT
PLAIN TEXT
00 20 31 D4 3E FF DA ..

Decryption Algorithm (cont):

- **Write decrypted text to output file**
- **Read/Write unencrypted**
  - **Until end of file**
  - **Unless another Huffman marker**

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- Future Considerations

Object Oriented Outline:

- Data Abstraction
  - ISE constructors
  - Virtual encrypt/decrypt methods
  - Data members and gets/sets
    - File names
    - Key
  - Make file name methods

Object Oriented Outline (cont):

- Information hiding
  - Data members
    - protected
  - Get/Set methods
    - File names
    - Key
    - File type

Object Oriented Outline (cont):

- Inheritance

JPEG_ISE Class
- Encrypt
- Decrypt

ISE Class
- Constructor
- Gets/Sets
- Data Members

Object Oriented Outline (cont):

- Polymorphism
  - Constructors
    - ise()
    - ise (key, input_file_name, ise_file_name)
      - encrypting
    - ise(key, ise_file_name, output_file_name)
      - decrypting

Object Oriented Outline (cont):

- Polymorphism
  - Encryption
    - encrypt_file()
    - encrypt_file(key, input_file_name, ise_file_name)
  - Decryption
    - decrypt_file()
    - decrypt_file(key, ise_file_name, output_file_name)

API Usage:

Encryption Scenario:

```
char[] myKey = “ISE_IS_THE_BEST”;
char[] myInputFile = “myImage.jpg”;
char[] myISEFile = “myImage.ise”;
jpeg_ise* myISE;
myISE = new jpeg_ise(myKey,myInputFile,MyISEFile);
myISE->encrypt_file();
delete myISE;
```

API Usage (cont):

Decryption Scenario:

```
char[] myKey = “ISE_IS_THE_BEST”;
char[] myISEFile = “myImage.ise”;
char[] myOutputFile = “myImageDecrypt.jpg”;
jpeg_ise* myISE;
myISE = new jpeg_ise();
myISE->set_key(myKey);
myISE->set_ise_file(myISEFile);
myISE->set_output_file(myOutputFile);
myISE->decrypt_file();
delete myISE;
```

OO Benefits:

- **Objects easily extendable to other formats**
- **Clean, reliable code**
- **Apply what we've learned**

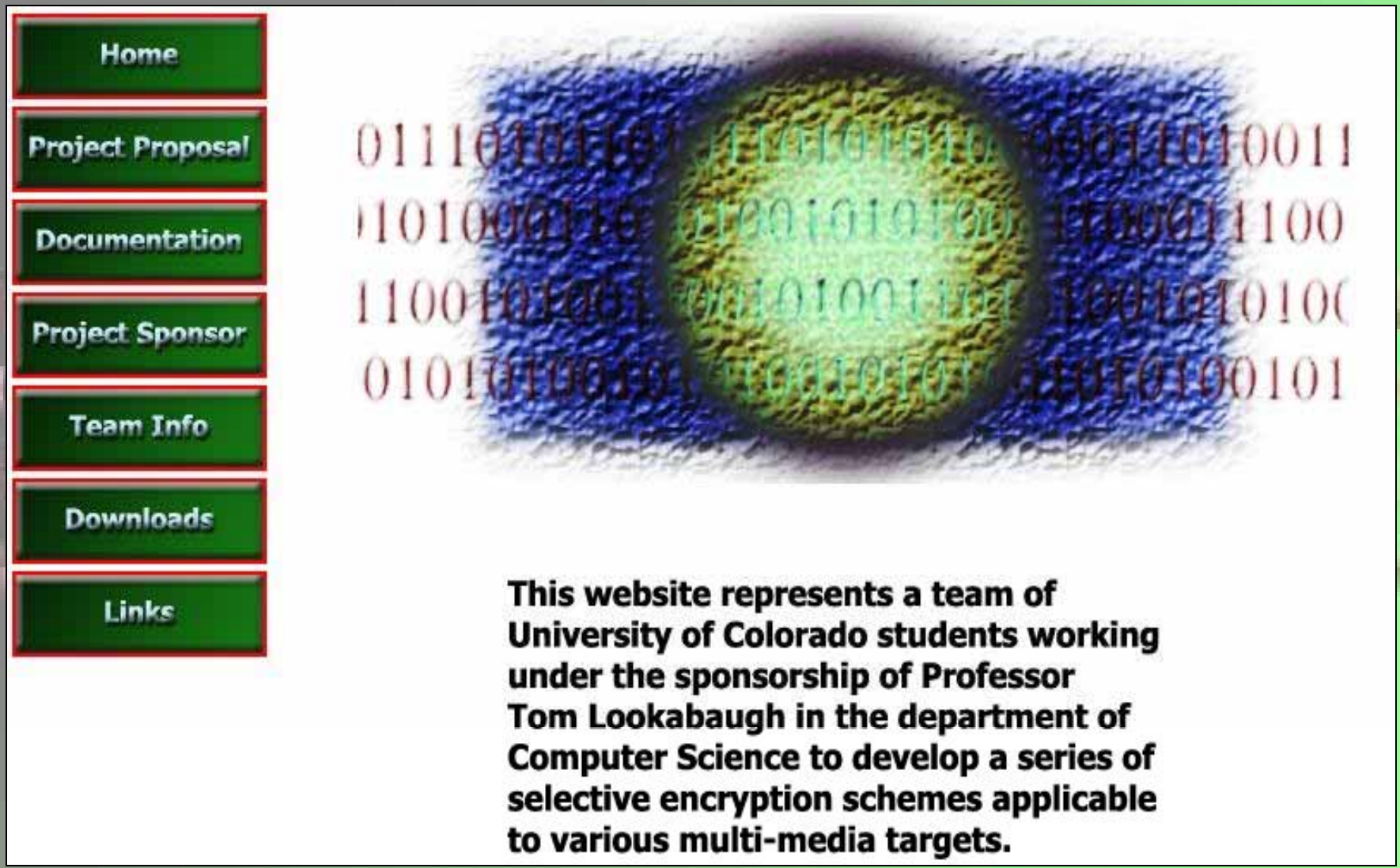
Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- **ISE Web Site Design**
- Future Considerations

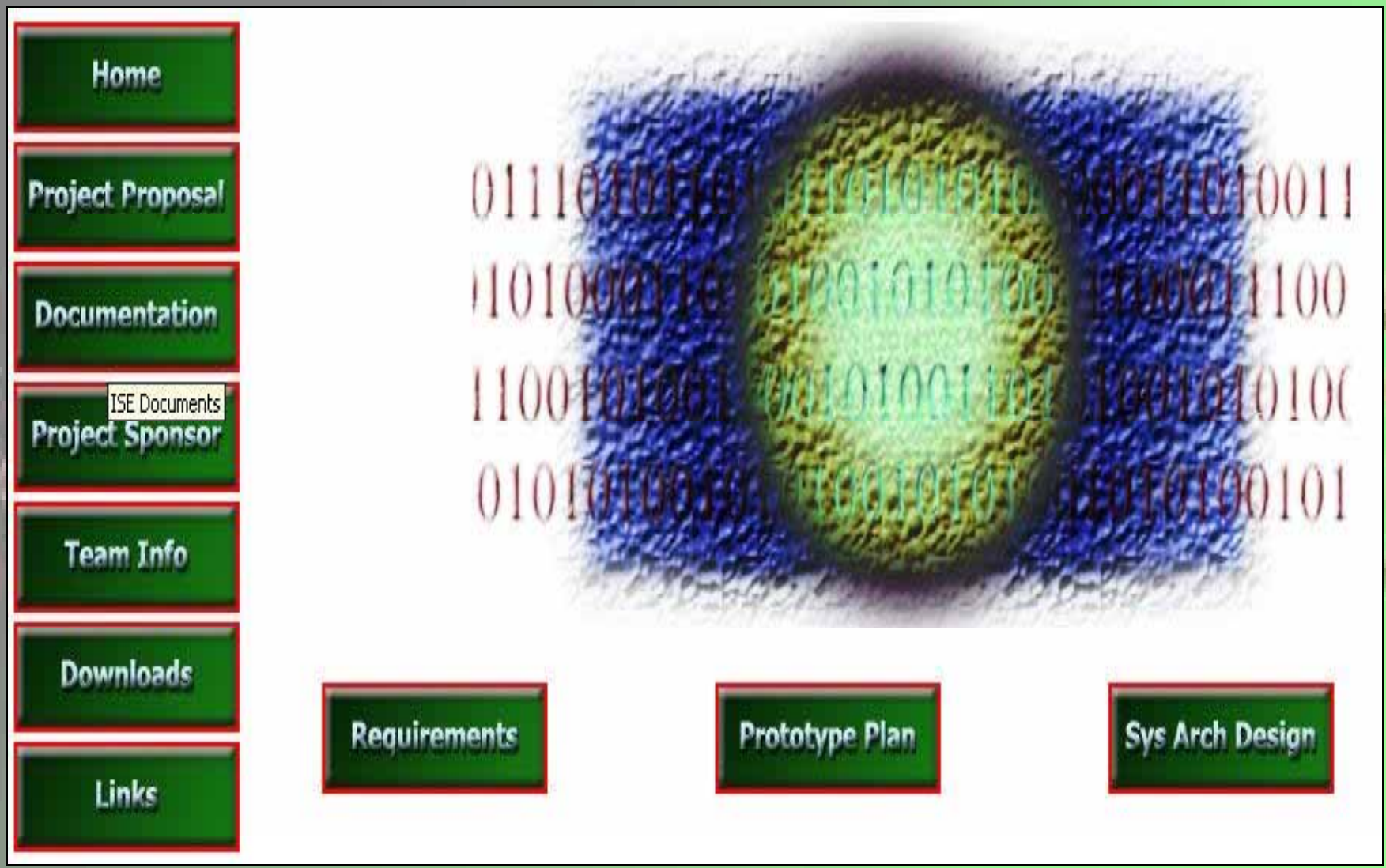
Requirements:

- **Easy to maintain**
- **Distribute products/documentation**
- **Create on existing computer in lab**
  - **http://128.138.75.184**

Home Page


Home
Project Proposal
Documentation
Project Sponsor
Team Info
Downloads
Links
This website represents a team of University of Colorado students working under the sponsorship of Professor Tom Lookabaugh in the department of Computer Science to develop a series of selective encryption schemes applicable to various multi-media targets.

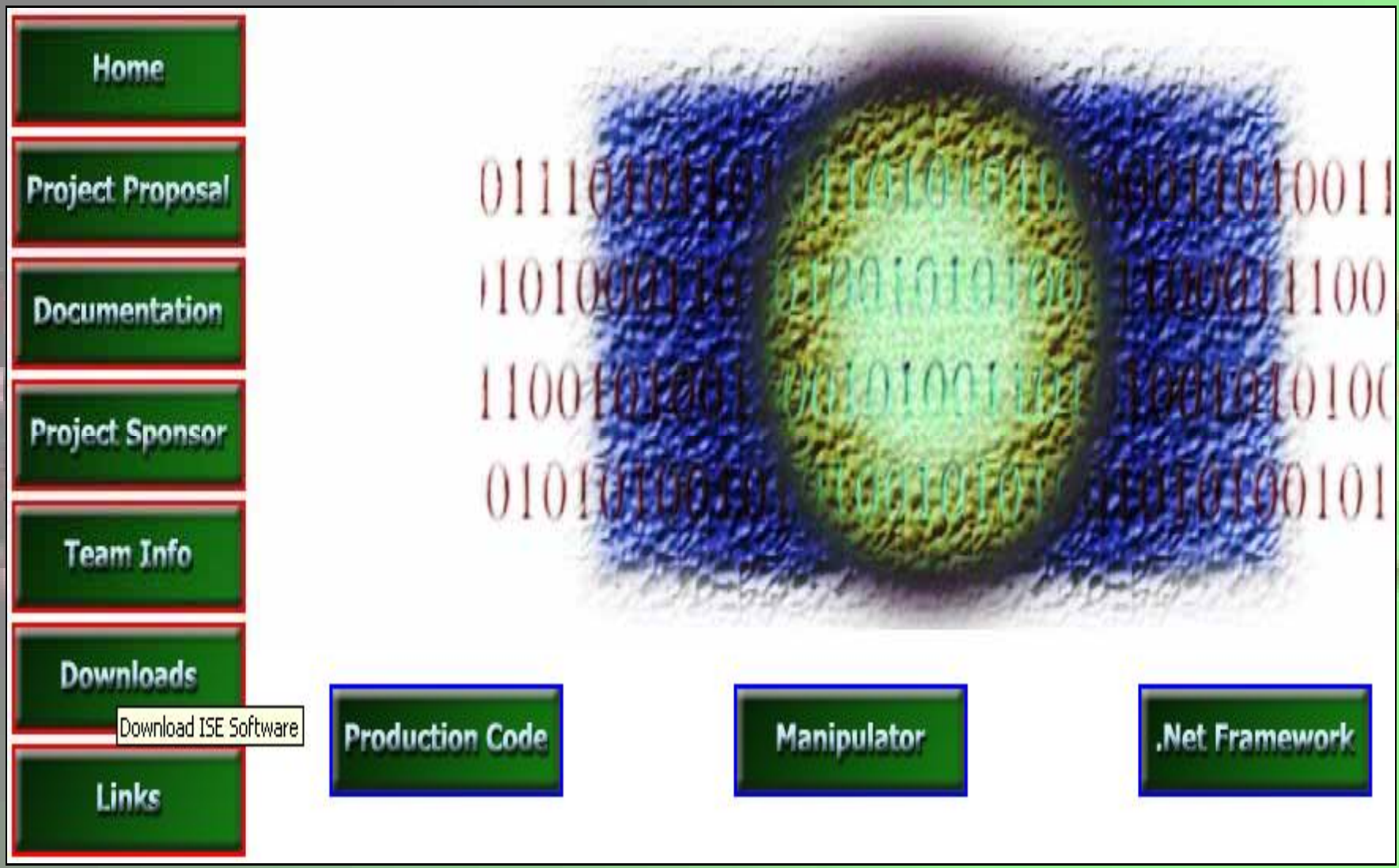
Documentation



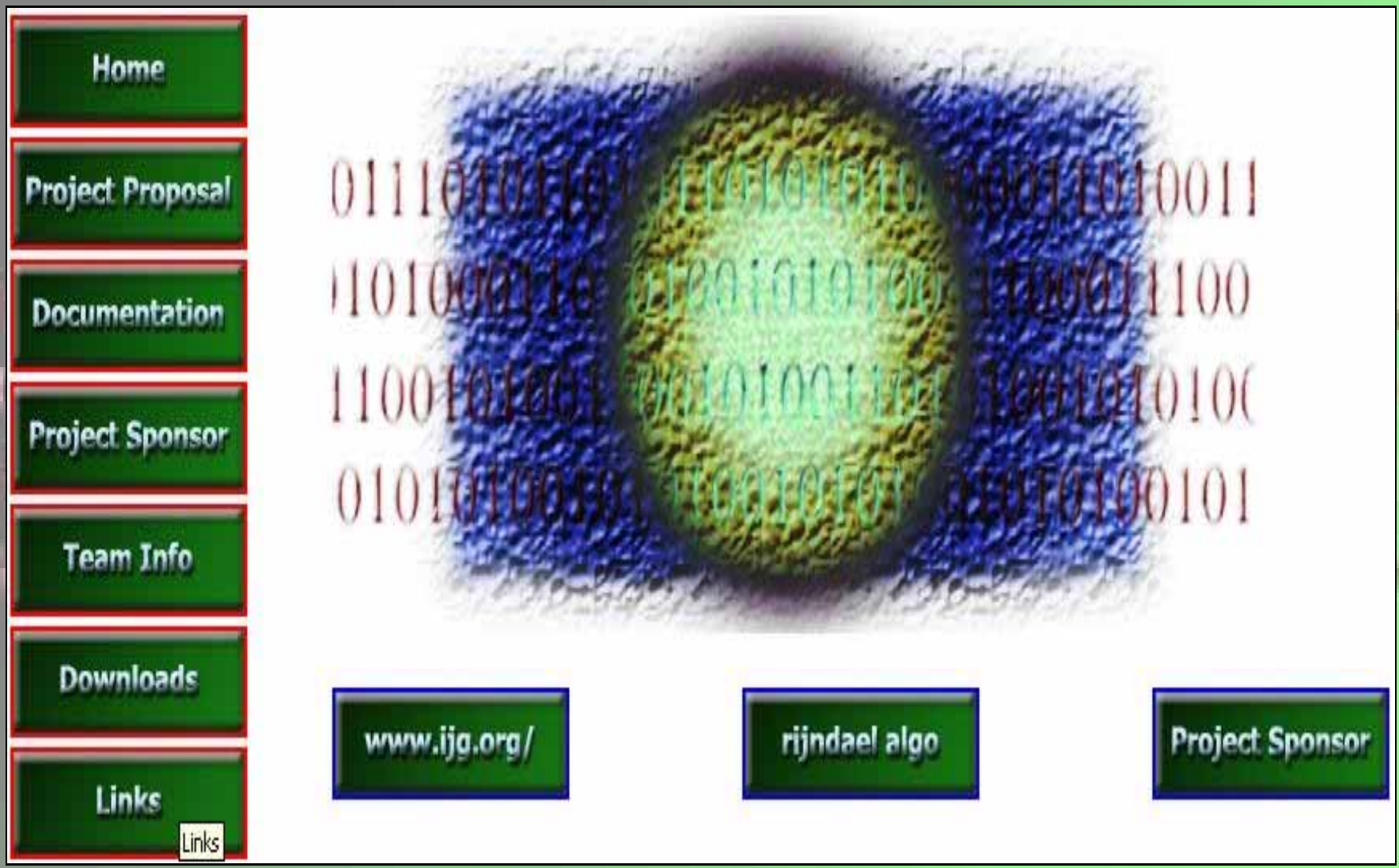
Web Site Design

Andrew

Downloads


Home
Project Proposal
Documentation
Project Sponsor
Team Info
Downloads
Download ISE Software
Links
Production Code
Manipulator
.Net Framework

Links


Home
Project Proposal
Documentation
Project Sponsor
Team Info
Downloads
Links
Links
www.ijg.org/
rijndael algo
Project Sponsor

Presentation Overview:

- Statement of problem
- Initial research into compressed files
- Target Selection Process
- JPEG Statistical Analysis
- JPEG Manipulator Design
- JPEG Manipulator Demonstration
- Encryption Algorithm Selection
- JPEG Selective Encryption Algorithms
- ISE Production Code Design
- ISE Web Site Design
- **Future Considerations**

Future Considerations:

- **Black hat attacks**
  - **Huffman table**
    - **Replacement**
    - **Reconstruction**
      - **Based on Quantizer**
      - **Based on Application**
  - **Quantizer table**
- **Publish web site for community**
- **Corrections**

Questions

Test Plan

Team ISE

Image Selective Encryption

ISE Test Plan

March, 2004

Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Project Proposal

Traffic constantly flows between computers connected to the Internet. Large volumes of information may take a long time traveling from destination to destination. Such a reduction in speed makes it desirable to compress the file as much as possible in order to send the smallest amount of data required. Thus, compression of data has allowed for the high-speed data transfers that have made Internet communication and business more feasible.

In addition to sending the smallest amount of information possible, users also attempt to maintain a certain level of security upon their information. Due to the fact that common encryption methods generally manipulate an entire file, most encryption algorithms tend to make the transfer of information more costly in terms of time and bandwidth. Thus, users pay a price for security relative to their desired level of security. One possible solution would be a system of encryption that works cooperatively with the standard compression schemes. *Selective Encryption* of only a small percentage of the file's bits will facilitate this solution. Because most encryption schemes will make the file larger, selective encryption seeks only to encrypt portions of the file that will make it unusable. In other words, if a user does not have the proper decryption device, the file should not be usable. Selective encryption will minimize the necessary increase in file size due to encryption while maintaining a maximum level of uselessness, or damage, to the product.

Team ISE (Image Selective Encryption) will deliver a package for selectively encrypting JPEG (Joint Photographic Experts Group) still image files. The package will provide the tools necessary to encrypt the critical information of a JPEG file in cooperation with existing standard compression tools. This package will handle JPEG files in such a way that only a small percentage of the total file will be encrypted. Selective Encryption security will not extend to the level of complete encryption, but rather to a level that would deter all but brute force attacks, allowing users to securely protect private JPEG images.

A JPEG image could be encrypted with any of the sufficiently secure encryption algorithms available to the open source community, but this can result in an increase in file size or can require a large amount of processing time. However, by selecting small but vital portions of a file and encrypting only those few bytes can render an image unusable. The initial statistical analysis done by the team will consist of specifically breaking down the standard JPEG compression scheme into its usable parts and evaluate which of the parts, if encrypted, will cause a potential user to pay for rights to the image or force subscription to the provider service.

An additional aspect of the encryption analysis will be the determination of the specific targets in the file for encryption. For example in an MPEG file there are headers that contain a small portion of the overall number of bits but which are extremely vital to the reproduction of the movie by the user. So, if certain headers were to be encrypted the percentage of the file being manipulated would be less than ten percent of the total number of bits in the file. Although only a small portion will be encrypted, the resulting damage experienced by an unauthorized user would be sufficient to cause the user to pay for the decryption package. However, there are other targets that, while they can be encrypted and will do sufficient damage, can be guessed by an

attacker. The attacker could, with some degree of effort, render the file useful without use of the decryption software. For example, if the frame rate of an MPEG file was encrypted, an attacker could try all three of most common frame rates and one of these is certain to produce the correct rate for the particular video. In the case of JPEG Selective Encryption, Team ISE will have to balance the targets for encryption against ease of simple attacks.

A permanent web site will be constructed by the team to make the software package available to anyone interested in the Team's project. As it is vital to the world of cryptography to let the community view the approach, the first form of the working prototype will be made available on the web site. From this, feedback can be received not only from the team itself, but also from the cryptography community at large.

So, following the guidelines of the ongoing MPEG research (also being guided by the sponsor), the team will study the JPEG process and earlier attempts at encryption. With the sponsor's assistance, Team ISE will devise a workable approach to handling individual JPEG images following the concept of selective encryption.

1. INTRODUCTION

Team ISE is sponsored by Assistant Professor of Computer Science, Tom Lookabaugh, at the University of Colorado: http://itd.colorado.edu/lookabaugh/. Tom Lookabaugh is currently involved in selective encryption research on standard MPEG (Moving Picture Experts Group) files and is interested in researching the application of Selective Encryption for other multimedia formats.

The goal of selective encryption is to minimize the amount of encryption applied to a file while maximizing the damage done to the image being viewed by a user not in possession of the authorized decryption package. Complete encryption is not a requirement of the process, nor is rendering the file useless to the level of complete military secrecy. It is acceptable for an attacker to be able to view portions of the file; however, the file should be distorted enough that an attacker would not wish to use the encrypted file, but would rather purchase or subscribe to the decryption method for access to the original files.

Multimedia files prove to be good subjects for selective encryption, as these files tend to be very large and employ compression algorithms that concentrate critical information in small portions of their bit stream. If the critical data in certain multimedia standards is encrypted properly, the remaining information becomes useless to those without the appropriate decryptor. There are many types of compression algorithms that fit this description, such as MPEG 1, 2 and 4 video, G.723 and G.729 video, AAC audio, MP3 audio, JPEG and JPEG2000 image formats. Applying a Selective Encryption security solution to selected multimedia formats will greatly increase the protection level of important information.

The focus of the ISE project is to research and develop an algorithm for selectively encrypting the JPEG *baseline* compression image standard. The product of the research and development will be a package that will encrypt a file so that the amount of the file being encrypted is relatively small (on the order of 1-2% of the total file). The product will be delivered in a package that will include an encryptor and a decryptor for JPEG files and a testing suite. A web site will be constructed to facilitate the delivery of the product and documentation about the process. The encryptor and decryptor will encrypt and decrypt selected targets contained within JPEG files. The ISE project will employ the AES (Advanced Encryption Standard) for our Selective Encryption algorithm. This package will be made available in a purely open source form on our final web site.

In addition to the package containing the decryptor and encryptor, Team ISE will also provide a test suite available to prospective users. The test suite will be used to aid in the research, development and testing of the team's final product. The test suite will provide the functions necessary to complete this project. First, it will allow the user to preview a standard JPEG image. Second, the test suite will break down the various portions of a JPEG image and provide the ability to manipulate the data in all of the portions. Third, after altering the data in any particular file, the test suite will provide the capability to preview the encryption attempt without the benefit of compatible decryption. Forth, the suite will have the ability to decrypt an encrypted file. The decryption options will allow the user try to defeat the encryption methods.

Any selective encryption scheme could be developed using a package that implemented these features, however, the delivered test suite will only employ the AES encryption scheme chosen by the team. The test suite will be available to download from the team web site.

The final web site will be deployed on a web server provided by the Sponsor. The machine facilitating the web server will use the Linux Red Hat 9.0 operating system platform. The team will acquire a fixed IP address from the proper University of Colorado authorities and will develop a simple web site capable of delivering information to viewers about the benefits and application of Selective Encryption technology. The site will provide users the option to download and use the final software package. The site will also provide links to important information and will remain in place as long as the sponsor deems necessary.

The final software package will accomplish the complex task of selectively encrypting a JPEG baseline standard image while providing a simple user interface. Team ISE has identified three specific types of users: high-end art users, typical Internet image users, and small, low-end image users. The research and software will be tailored to these users' needs. Figure 1.1 is a flow chart showing the general logic design of the team's final product.

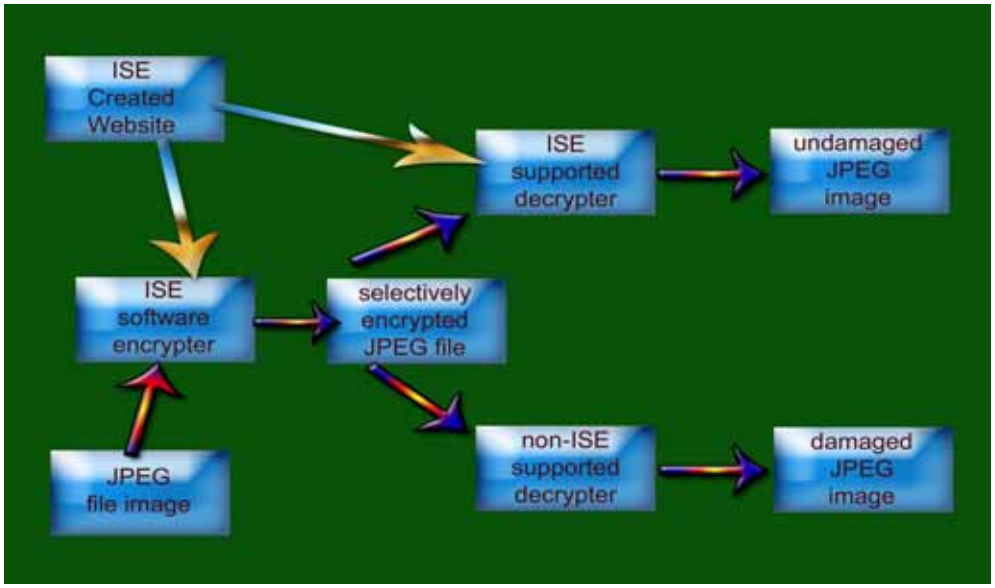


Figure 1.1: Conceptual Overview of ISE Software

This document describes the test plan for the various components of the ISE project, and is used to verify that the project meets the requirements set forth in the ISE *Requirements Document*. It describes the environments, both hardware and software, necessary to test the production code, Manipulator, and web site. It then proceeds to give a detailed description of the tests themselves.

2. TEST ENVIRONMENT

This section of the text plan document outlines that Environment used to test the ISE Production Code, the ISE Manipulator, and the ISE web site. The ISE Production Code tests should be conducted in the following environment:

Software:

- Any Version of Redhat Linux 9.0 and higher.
- Windows 9x/ME/NT/200x/XP and higher.
- Mac OS X or higher.

Hardware:

- Generic Color Monitor.
- Mouse as part of the User Interface.
- Keyboard as part of the User Interface.
- Support for a 32-bit processor assembly instructions for AES optimizations.

The ISE Manipulator tests should be conducted in the following environment:

Software:

- Windows 9x/ME/NT/200x/XP and higher.
- Microsoft .NET Framework Version 1.1 or Higher.

Hardware:

- Generic Color Monitor.
- Mouse as part of the User Interface.
- Keyboard as part of the User Interface.

The ISE Web Site tests should be conducted in the following environment:

Software:

- Microsoft Internet Explorer 6.0 or higher.
- Netscape Navigator 6.0 or higher.
- Mozilla 1.5.1 or higher.
- Safari 1.0 or higher.
- Support for HTML version 4.01 transitional.

Hardware:

- Generic Color Monitor.
- Mouse as part of the User Interface.
- Keyboard as part of the User Interface.

Unless explicitly invoking any instance of the ISE products as part of a test procedure, these tests assume that an instance of the product is running.

3. TESTS

The tests are organized into three separate sections which deal with the different components of the ISE project. The sections are:

1. **ISE Production Code**
2. **ISE Manipulator**
3. **ISE Web Site**

Each test in the Test Plan has seven components:

Purpose	The reason for the test.
Procedure	The steps to follow to conduct the test.
Expected Result	The results necessary to pass the test.
Comments	Any comments the tester might have.
Date	Date the test was conducted.
Tester	Name of the person conducting the test.
Outcome	Outcome of the test (Pass or Fail).

3.1. Production Code Test

This section of the test plan is to outline out all of the testing requirements and desired results for the ISE Production Code. To test all of the functionality provided by this class, we've have designed a set of tests to cover all of the class methods. These tests were conducted as outlined in the following sections.

3.1.1. JPEG_ISE Constructor with Key Only

Purpose:	The purpose of this test is to determine if a jpeg_ise object can be created with only an encryption/ decryption key.
Procedure:	1. Create a pointer to a character array in a C++ program containing the desired key information. 2. Call the jpeg_ise(key) constructor with this key as the only parameter.
Expected Result:	A new object of type jpeg_ise will be created. The key will be set using the information passed in the parameter. A default value of NULL will be set for both the input and output file names.
Comments:	In order to use this object for encryption or decryption, the user must call the set_input_file_name() and set_output_file_name() functions to set the desired jpeg and ise files.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.2. JPEG_ISE Constructor with All Parameters

Purpose: The purpose of this test is to determine if a jpeg_ise object can be created with an encryption/ decryption key as well as the input and/or output file name.

Procedure: 1. Create three pointers to character arrays in a C++ program, the first containing the desired key information, the second containing the input file name and the output file name.
2. Call the jpeg_ise() constructor with all three pointers as it's arguments.

Expected Result: A new object of type jpeg_ise will be created. The key will be set using the information passed in the first parameter. The second and third parameters will be used to set the input and output file names.

Comments: For the input and output file names, one parameter should be a jpeg file name and the other should be an ise file name, in either order. The user can verify that the input and output files were set correctly using the get_input_file_name() and get_output_file_name() functions.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: **Pass**

3.1.3. Set_Key Function with Valid Key

Purpose: The purpose of this test is to determine if a key can be created with a valid character string.

Procedure: 1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing one or more characters indicating the desired key information.
3. Call the set_key() function with this key as the only parameter.

Expected Result: The encryption/decryption key will be created for the object using the information in the character array. The function should return 0 to indicate that the key was successfully created for the object.

Comments: The key information in the calling program should not be damaged or modified in any way by this function.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.4. Set_Key Function with Invalid Key

Purpose: The purpose of this test is to determine if the set_key() function exits gracefully given NULL for the key information.

Procedure: 1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing NULL, which is invalid for jpeg_ise key information.
3. Call the set_key() function with this key as the only parameter.

Expected Result: It should return 1 indicating an invalid key.

Comments: If the object did not contain a valid key previous to this function call, the function will need to be called again with a valid key for the object to be used for encryption or decryption.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.5. Set_Input_File_Name Function with Valid Input File

Purpose: The purpose of this test is to determine if an input file name can be created with a valid character string.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing the desired input file name with a .jpeg, .jpg, or .ise extension.
3. Call the set_input_file_name() function with this pointer as the only parameter

Expected Result: The input file name will be created for the object using the information in the character array. The function should return 0 to indicate that the input file name was successfully created for the object.

Comments: The input file name information in the calling program should not be damaged or modified in any way by this function. The input file must exist and be of either ise or jpeg type.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.6. Set_Input_File_Name Function with NULL

Purpose: The purpose of this test is to determine if the set_input_file_name() function exits gracefully given NULL for the file name.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing NULL for the input file name.
3. Call the set_input_file_name() function with this pointer as the only parameter.

Expected Result: The function should exit without setting the jpeg_ise object's input file name. It should return 1 indicating an invalid file name.

Comments: If the object did not contain a valid input file name previous to this function call, the function will need to be called again with a valid file name for the object to be used for encryption or decryption.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.7. Set_Input_File_Name Function with Non-Valid File

Purpose: The purpose of this test is to determine if the set_input_file_name() function exits gracefully given a non-valid file for the file name, i.e. the file is of neither jpeg nor ise type.

Procedure

1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing a non-valid file for the input file name. Examples of non-valid files are bitmaps, text files, or any other non-jpeg or non-ise file types.
3. Call the set_input_file_name() function with this pointer as the only parameter.

Expected Result: The function should exit without setting the jpeg_ise object's input file name. It should return 1 indicating an invalid file name.

Comments: If the object did not contain a valid input file name previous to this function call, the function will need to be called again with a valid file name for the object to be used for encryption or decryption.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.8. Set_Output_File_Name Function with Valid Output File

Purpose: The purpose of this test is to determine if an output file name can be created with a valid character string.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing the desired output file name with a .jpeg, .jpg, or .ise extension.
3. Call the set_output_file_name() function with this pointer as the only parameter.

Expected Result: The output file name will be created for the object using the information in the character array. The function should return 0 to indicate that the output file name was successfully created for the object.

Comments: The output file name information in the calling program should not be damaged or modified in any way by this function. The output file must exist and be of either ise or jpeg type.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.9. Set_Output_File_Name Function with NULL

Purpose: The purpose of this test is to determine if the set_output_file_name() function exits gracefully given NULL for the file name.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing NULL for the output file name.
3. Call the set_output_file_name() function with this pointer as the only parameter.

Expected Result: The function should exit without setting the jpeg_ise object's output file name. It should return 1 indicating an invalid file name.

Comments: If the object did not contain a valid output file name previous to this function call, a default name will be created during encryption or decryption based on the input file name.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.10. Set_Output_File_Name Function with Non-Valid File

Purpose: The purpose of this test is to determine if the set_output_file_name() function exits gracefully given a non-valid file for the file name, i.e. the file is of neither jpeg nor ise type.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing a non-valid file for the output file name. Examples of non-valid files are bitmaps, text files, or any other non-jpeg or non-ise file types.
3. Call the set_output_file_name() function with this pointer as the only parameter.

Expected Result: The function should exit without setting the jpeg_ise object's output file name. It should return 1 indicating an invalid file name.

Comments: If the object did not contain a valid output file name previous to this function call, a default name will be created during encryption or decryption based on the input file name.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.11. Get_Input_File_Name Function When input_file_name != NULL

Purpose: The purpose of this test is to determine if the get_input_file_name() function returns the proper string indicating the name of the input file.

Procedure: 1. Create a jpeg_ise object with a valid input file.
2. Call the get_input_file_name() function with no parameters.

Expected Result: The function should return a pointer to a character string containing the same name as the input file used when creating the object.

Comments: If the input file is properly set for the jpeg_ise object, then a valid pointer to the char array will be returned.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.12. Get_Input_File_Name Function When input_file_name == NULL

Purpose: The purpose of this test is to determine if the get_input_file_name() function returns NULL when the input_file_name is equal to NULL.

Procedure: 1. Create a jpeg_ise with key only.
2. Call the get_input_file_name() function with no parameters.

Expected Result: The function should return NULL.

Comments: If the input file is not explicitly set by the user for the jpeg_ise object, then the default NULL will be returned.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.13. Get_Output_File_Name Function When input_file_name != NULL

Purpose: The purpose of this test is to determine if the get_output_file_name() function returns the proper string indicating the name of the output file.

Procedure:
1. Create a jpeg_ise object with a valid output file.
2. Call the get_output_file_name() function with no parameters.

Expected Result: The function should return a pointer to a character string containing the same name as the input file used when creating the object.

Comments: If the output file is properly set for the jpeg_ise object, then a valid pointer to the char array will be returned.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.14. Get_Output_File_Name Function When input_file_name == NULL

Purpose: The purpose of this test is to determine if the get_output_file_name() function returns NULL when the input_file_name is equal to NULL.

Procedure:
1. Create a jpeg_ise object with key only.
2. Call the get_output_file_name() function with no parameters.

Expected Result: The function should return NULL.

Comments: If the output file is not explicitly set by the user for the jpeg_ise object, then the default NULL will be returned.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.15. Encrypt_File Function Normal Use

Purpose: The purpose of this test is to determine if the encrypt_file() function selectively encrypts a jpeg image.

Procedure:
1. Create a jpeg_ise object with a valid key, input jpeg file, and output ise file.
2. Call the encrypt_file() function with no parameters.

Expected Result: The function should return 0 to indicate success. The original jpeg image should be undamaged and the ISE file should contain the encrypted jpeg.

Comments: The function should return 0 to indicate success. The original jpeg image should be undamaged and the ise file should contain the encrypted jpeg.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.16. Encrypt_File Function with Invalid Input File

Purpose: The purpose of this test is to determine if the encrypt_file() function exits gracefully given an input file that does not exist.

Procedure:
1. Create a jpeg_ise object with a valid key and output ise file name and a jpeg file name that does not exist.
2. Call the encrypt_file() function with no parameters.

Expected Result: The function should return 1 to indicate that the input jpeg file could not be opened. The function should then exit without encrypting any data.

Comments: The output ise file should be empty due to the fact that no encryption was performed.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.17. Encrypt_File Function with Output ISE File Name Not Set

Purpose: The purpose of this test is to determine if the encrypt_file() function calls the make_ise_file_name() function to make a default output ise file.

Procedure: 1. Create a jpeg_ise object with a valid key and input jpeg file. Leave the output file name to be the default NULL.
2. Call the encrypt_file() function with no parameters.

Expected Result: The function should call make_ise_file_name() to create an ise file name based on the input jpeg file name. Encryption should proceed and return 0 indicating a success.

Comments: The output ise file should be created and named based on the input jpeg file. This file will contain the encrypted jpeg file information. If the ise file could not be created for any reason, this function will return 2.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.18. Decrypt_File Function Normal Use

Purpose: The purpose of this test is to determine if the decrypt_file() function selectively decrypts an ise image.

Procedure: 1. Create a jpeg_ise object with a valid key, input ise file, and output jpeg file.
2. Call the decrypt_file() function with no parameters.

Expected Result: The function should return 0 to indicate success. The original ise image should be undamaged and the new jpeg file should contain the exact same information as the original jpeg.

Comments: To test if the image decrypted properly, the user can try to look at the image. Also, to make sure that there is no difference between the original and decrypted jpeg images, the user could run the Unix “diff” command on the two files.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.19. Decrypt_File Function with Non-JPEG-ISE Input File

Purpose: The purpose of this test is to determine if the decrypt_file() function exits gracefully given an input ise file that is not an encrypted jpeg image, i.e. the ise file contains a decrypted mp3 or zip file.

Procedure:
1. Create a jpeg_ise object with a valid key and output jpeg file name and an ise file name that contains an encrypted mp3 or zip file.
2. Call the decrypt_file() function with no parameters.

Expected Result: The function should return 5 to indicate that the input file is not jpeg-ise. The function should then exit without decrypting any data.

Comments: Due to the fact that the only ise files that exist are all from jpegs, the tester will have to change the first byte in the ise to mimic a different ise file type.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.20. Decrypt_File Function with Invalid Input File

Purpose: The purpose of this test is to determine if the decrypt_file() function exits gracefully given an input ise file that does not exist.

Procedure:
1. Create a jpeg_ise object with a valid key and output jpeg file name and an ise file name that does not exist.
2. Call the decrypt_file() function with no parameters.

Expected Result: The function should return 2 to indicate that the input ise file could not be opened. The function should then exit without decrypting any data.

Comments: The output jpeg file should be empty due to the fact that no decryption was performed.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.21. Decrypt_File Function with Output File Name Not Set

Purpose: The purpose of this test is to determine if the decrypt_file() function calls the make_output_file_name() function to make a default output file.

Procedure:
1. Create a jpeg_ise object with a valid key and input ise file. Leave the output file name to be the default NULL.
2. Call the decrypt_file() function with no parameters.

Expected Result: The function should call make_output_file_name() to create an output file name based on the input ise file name. Decryption should proceed and return 0 indicating a success.

Comments: The output jpeg file should be created and named based on the input jpeg file. This file will contain the decrypted jpeg file information. If the ise file could not be created for any reason, this function will return 2.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.22. Decrypt_File Function with Incorrect Key

Purpose The purpose of this test is to make sure that the decrypt_file() function does not produce a properly decrypted jpeg image when given an incorrect key.

Procedure:	1. Encrypt a jpeg image and create an ise file with a valid key 2. Call the set_key() function with a new valid key. 3. Call decrypt_file with the ise file and the new key.
Expected Result:	The function should return 0 to indicate that the file was decrypted. The new jpeg image produced should not be a valid jpeg image.
Comments:	To test that the image did not decrypted properly, the user can try to look at the new image. Also, the user could run the Unix "diff" command on the original image and the new decrypted image to see that there are differences.
Date:	March 6, 2004
Tester:	Joe Jarchow / Joseph Kadhim / Shinya Daigaku
Outcome:	Pass

3.2. Manipulator Test

This section of the test plan is to list out all of the testing requirements and desired results for the ISE JPEG Manipulator. To test this massive amount of functionality, this testing breaks down into three main pieces:

1. Menu Options
2. Button Controls
3. General Tests

The "Menu Options" section will test all of the different menu options available in the Manipulator, like the "Save Project" or "Open Picture" options that are available. The "Button Controls" section will test all of the different button control found within the Manipulator, like "Save Project" or "Update Picture" buttons available on the Project sub-tab on the Console tab. Finally, the "General Tests" section of this document will test all the rest of the miscellaneous functionality, like if the SEP project file is set up correctly or to test if the TextBox controls are working correctly.

3.2.1. Menu Options

This section of the test plan is to list out the menu functions that need to be tested. Included in this section is each of the tests, a short description of the test and the expected results.

3.2.1.1. File Menu Tests

This section of the test plan is to test all of the File Menu options. Each of the File Menu options has a test under this section.

3.2.1.1.1. New Project Menu Option Test

Purpose: To test the "New Project" menu option to make sure that a new project is created when this option is selected.

Procedure: 1. Prior to choosing the "New Project" option, open a new picture in the Manipulator.
2. Then click the "New Project" menu option under the File menu.

Expected Result: All of the old information in the Manipulator should be cleared out for a new project to be created and they should be prompted for a new project file name and path.

Comments: This is not required to make a new project, for instance, you could just load in a picture and then click the "Save Project" option and the current information loaded into the Manipulator will be saved. This option is intended to allow the user to quickly clear out the Manipulator and start a new project.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.1.2. Open Project Menu Option Test

Purpose: To test the "Open Project" menu option to make sure that a previously saved project is loaded into the Manipulator when this option is selected.

Procedure: 1. Prior to choosing the "Open Project" option, open a new picture in the Manipulator.
2. Change a few values in some of the text controls.
3. Then click the "Open Project" menu option under the File menu.
4. Choose a valid SEP project file to be loaded by using the dialog box.

Expected Result: When the "Open Project" option is selected, the user should be prompted to first save any previous information. Then, all of the old information loaded in the Manipulator should be cleared out for a project being loaded and then all previous project information should be reloaded properly.

Comments: This test should probably be completed in conjunction with the next test, which is the "Save Project" option.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.1.3. Save Project Menu Option Test

Purpose: To test the "Save Project" menu option to make sure that a project is saved properly in the SEP file to be stored for future use.

Procedure: 1. Prior to choosing the "Save Project" option, open a new picture in the Manipulator.
2. Change a few values in some of the text controls.
3. Then click the "Save Project" menu option under the File menu.
4. Choose a valid name and file path for the SEP project file to be created.

Expected Result: When the "Save Project" option is selected, the user should be prompted to choose a location and file name for the SEP project file. Then, all of the current information loaded in the Manipulator should be saved in the project file being created.

Comments: This test should probably be completed in conjunction with the next test, which is the "Open Project" menu option.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.1.4. Open Picture Menu Option Test

Purpose: To test the "Open Picture" menu option to make sure that a picture and its data are properly loaded into the Manipulator.

Procedure: 1. Have a valid JPEG image and an invalid JPEG image available.
2. Try opening both, one at a time, in the Manipulator

Expected Result: The valid JPEG should be loaded into the Manipulator with all the values loaded into the interface, under the proper headings. The invalid image load attempt should generate an error message about the file structure.

Comments: The Manipulator should not discriminate against file name, but rather the file structure. Even if the file is a valid JPEG but labeled as .BMP or some other format, the file should still load properly.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.1.5. Update Picture Menu Option Test

Purpose: To test the "Update Picture" menu option to make sure that a picture is generated from the values that are currently loaded in the Manipulator interface (whether they are user updated or not).

Procedure: 1. Load a picture into the Manipulator.
2. Before changing any values, try making a replica of the picture by choosing the "Update Picture" menu option.
3. Then try changing some values in the Manipulator and choose the "Update Picture" option again.
4. Using the converted program, convert all 3 files (the original and the 2 new images) and verify that both the files were created with information provided in the Manipulator.

Expected Result: The first picture created should have the exact same values as original converted picture. The second picture should only have values that are different from the original where they were changed/updated in the Manipulator.

Comments: Only change a few values at first to changed to make sure they work properly for all the fields.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.2.1.1.6. Exit Menu Option Test

Purpose: To test the "Exit" menu option to make sure that a the user can properly exit the program.

Procedure:
1. Load a picture into the Manipulator.
2. Change a few values, but don't do anything else.
3. Be sure NOT to save before hitting the "Exit" option.
4. Choose the "Exit" menu option.

Expected Result: Before the application is closed, the user should be prompted to save the current information. Then, after the user has provided input, the application should be closed.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.2 Edit Menu Tests

This section of the test plan is to test all of the Edit Menu options. Each of the Edit Menu options has a test under this section.

3.2.1.2.1. Copy Menu Option Test

Purpose: To test the "Copy" menu option to make sure that when text is selected, we copy it to the system clipboard.

Procedure:
1. Load a picture into the Manipulator.
2. Highlight some data values.

3. Click the “Copy” menu option.
4. In some other program, like notepad or word, try pasting the text in.

Expected Result: The highlighted text in the Manipulator should be pasted to the new document. Also, the text in the Manipulator should remain unchanged.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.2.2. Cut Menu Option Test

Purpose: To test the “Cut” menu option to make sure that the user can cut text out of a given field and paste it back into another.

Procedure:
1. Load a picture into the Manipulator.
2. Highlight some data values.
3. Click the “Cut” menu option.
4. In some other program, like notepad or word, try pasting the text in.

Expected Result: The highlighted text in the Manipulator should be pasted to the new document. Also, the text in the Manipulator should be removed from the text control.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.2.3. Paste Menu Option Test

Purpose: To test the “Paste” menu option to make sure that the user can paste text into the different text controls in the Manipulator.

Procedure: 1. Load a picture into the Manipulator.
2. Highlight some data values.
3. Click the “Copy” menu option.
4. Highlight some other data values.
5. Click the “Paste” menu option.

Expected Result: The highlighted text in the Manipulator should be pasted to the selected text.

Comments: If for some reason the “Copy” menu won’t work, use <ctrl+c> button, which is guaranteed to work.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3. View Menu Tests

This section of the test plan is to test all of the View Menu options. Each of the View Menu options has a test under this section.

3.2.1.3.1. Stretch Large Original Menu Option Test

Purpose: To test the “Stretch Large Original” menu option to make sure that the user can both stretch and view normally the large original picture on the Original Picture tab.

Procedure: 1. Load a picture into the Manipulator.
2. Click on the Original Picture tab.
3. Click the “Stretch Large Original” menu option several times.

Expected Result: The Large Original image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3.2. Stretch Large Changed Menu Option Test

Purpose: To test the "Stretch Large Changed" menu option to make sure that the user can both stretch and view normally the large changed picture on the Changed Picture tab.

Procedure: 1. Load a picture into the Manipulator.
2. Click on the Changed Picture tab.
3. Click the "Stretch Large Changed" menu option several times.

Expected Result: The Large Changed image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3.3. Stretch Small Original Menu Option Test

Purpose: To test the "Stretch Small Original" menu option to make sure that the user can both stretch and view normally the small original picture on the Console tab.

Procedure: 1. Load a picture into the Manipulator.
2. Click on the Console Picture tab.
3. Click the "Stretch Small Original" menu option several times.

Expected Result: The Small Original image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3.4. Stretch Small Changed Menu Option Test

Purpose: To test the "Stretch Small Changed" menu option to make sure that the user can both stretch and view normally the small changed picture on the Console tab.

Procedure:
1. Load a picture into the Manipulator.
2. Click on the Console Picture tab.
3. Click the "Stretch Small Changed" menu option several times.

Expected Result: The Small Changed image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3.5. Stretch All Menu Option Test

Purpose: To test the "Stretch All" menu option to make sure that the user can both stretch and view normally all of the images in one click.

Procedure:
1. Load a picture into the Manipulator.
2. Click the "Stretch All" menu option several times.
3. Each time you click the "Stretch All" option be sure to check all of the pictures to make sure they updated correctly.

Expected Result: The Small Changed image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.4. About Menu Tests

This section of the test plan is to test all of the About Menu options. Each of the About Menu options has a test under this section.

3.2.1.4.1. About Menu Option Test

Purpose: To test the "About" menu option to make sure that the user can view the project information and the people associated with the project.

Procedure: 1. Click the "About" menu option.

Expected Result: A new window should open up with the appropriate project information. This window should close when is it clicked on.

Comments: Try several times in a row to make sure it works right.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2. Button Control Tests

This section of the test plan is to list out the menu functions that need to be tested. Included in this section is each of the tests, a short description of the test and the expected results.

3.2.2.1. Project Sub-Tab Button Tests

This section of the test plan is to test all of the buttons on the Project sub-tab. Each of the buttons on the Project sub-tab has a test under this section.

3.2.2.1.1. New Project Button Test

Purpose: To test the "New Project" button to make sure that a new project is created when this option is selected.

Procedure: 1. Prior to clicking the "New Project" button, open a new picture in the Manipulator.
2. Then click the "New Project" button under the Project sub-tab on the Console tab.

Expected Result: All of the old information in the Manipulator should be cleared out for a new project to be created and they should be prompted for a new project file name and path.

Comments: This is not required to make a new project, for instance, you could just load in a picture and then click the "Save Project" button and the current information loaded into the Manipulator will be saved. This option is intended to allow the user to quickly clear out the Manipulator and start a new project.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.2. Load Project Button Test

Purpose: To test the "Load Project" menu option to make sure that a previously saved project is loaded into the Manipulator when this option is selected.

Procedure: 1. Prior to clicking the "Load Project" button, open a new picture in the Manipulator.
2. Change a few values in some of the text controls.
3. Then click the "Load Project" button located under the Project sub-tab under the Console tab.
4. Choose a valid SEP project file to be loaded by using the dialog box.

Expected Result: When the "Load Project" button is clicked, the user should be prompted to first save any previous information. Then, all of the old information loaded in the Manipulator should be cleared out for a project being loaded and then all previous project information should be reloaded properly.

Comments: This test should probably be completed in conjunction with the next test, which is the "Save Project" button.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.3. Save Project Button Test

Purpose: To test the "Save Project" button to make sure that a project is saved properly in the SEP file to be stored for future use.

Procedure: 1. Prior to clicking the "Save Project" button, open a new picture in the Manipulator.
2. Change a few values in some of the text controls.
3. Then click the "Save Project" button under the Project sub-tab under the Console.
4. Choose a valid name and file path for the SEP project file to be created.

Expected Result: When the "Save Project" button is clicked, the user should be prompted to choose a location and file name for the SEP project file. Then, all of the current information loaded in the Manipulator should be saved in the project file being created.

Comments: This test should probably be completed in conjunction with the next test, which is the "Open Project" button.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.4. Load Picture Button Test

Purpose: To test the "Load Picture" button to make sure that a picture and its data are properly loaded into the Manipulator.

Procedure: 1. Have a valid JPEG image and an invalid JPEG image available.
2. Try opening both, one at a time, in the Manipulator by clicking on the "Load Picture" button.

Expected Result: The valid JPEG should be loaded into the Manipulator with all the values loaded into the interface, under the proper headings. The invalid image load attempt should generate an error message about the file structure.

Comments: The Manipulator should not discriminate against file name, but rather the file structure. Even if the file is a valid JPEG but

labeled as .BMP or some other format, the file should still load properly.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.5. Save Picture Button Test

Purpose: To test the "Save Picture" button to make sure that a picture load in the changed picture image boxes are properly saved to file as a JPEG image.

Procedure: 1. Open a valid JPEG image in the Manipulator.
2. Alter a few values and create an image that is not the same, but still viewable as a JPEG image.
3. Click the "Save Picture" button located on the Project sub-tab of the Console tab.

Expected Result: The viewable JPEG loaded into the Manipulator changed image boxes should be saved to file. The values saved should be the values that are currently loaded into the text controls (except the encoded stream) of the Manipulator.

Comments: This image should be tested by trying to open the created JPEG image in a standard image viewer (or multiple viewers for that matter). This image should be viewable as normal.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.6. Update Picture Button Test

Purpose: To test the "Update Picture" button to make sure that a picture is generated from the values that are currently loaded in the Manipulator interface (whether they are user updated or not).

Procedure: 1. Load a picture into the Manipulator.
2. Before changing any values, try making a replica of the picture by clicking the "Update Picture" button.
3. Then try changing some values in the Manipulator and click the "Update Picture" button again.
4. Using the converted program, convert all 3 files (the original and the 2 new images) and verify that both the files were created with information provided in the Manipulator.

Expected Result: The first picture created should have the exact same values as original converted picture. The second picture should only have values that are different from the original where they were changed/updated in the Manipulator.

Comments: Only change a few values at first to changed to make sure they work properly for all the fields. You may want to use the converter to convert the images produced hexadecimal to evaluate the data contained in the image.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.2. Huffman and Quantizer Sub-Tab Button Tests

This section of the test plan is to test all of the buttons on the Project sub-tab. Each of the buttons on the Project sub-tab has a test under this section.

3.2.2.2.1. Clear Button Tests

Purpose: To test the all "Clear" buttons on their corresponding text controls on both of the Huffman sub-tabs and the Quantizer sub-tab.

Procedure: 1. Prior to clicking the "clear" button, open a new picture in the Manipulator.
2. Try altering text in each for the Huffman tables and Quantizer tables.
3. Then, for each of the different Quantizer and Huffman table fields, click the corresponding clear button.

Expected Result: The corresponding table field should be cleared out. Also, check to make sure that click one clear doesn't affect any of the other text fields.

Comments: Most images won't have 4 Quantizer and/or 8 Huffman tables, so to test the unused fields, simply type some data into the field and then hit the "Clear" button. Also, if the field hasn't been previously altered, then it original data should be moved to the corresponding original data field.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.2.2. Random Button Tests

Purpose: To test the all "Random" buttons on their corresponding text controls on both of the Huffman sub-tabs and the Quantizer sub-tab.

Procedure: 1. Prior to clicking the "Random" button, open a new picture in the Manipulator.
2. Try clicking the "Random" button to add a random byte onto the end of the tables.

Expected Result: The corresponding table field should have a random byte appended to the end of it. Also, make sure that if the field has not been altered previously, that the information be moved to the corresponding original data TextBox control.

Comments: Most images won't have 4 Quantizer and/or 8 Huffman tables, so to test the unused fields, simply hit the "Random" button, a byte will still be added to the end of an empty table. Also, if the field hasn't been previously altered, then it original data should be moved to the corresponding original data field.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.2.3. Restore Button Tests

Purpose: To test the all “Restore” buttons on their corresponding text controls on both of the Huffman sub-tabs and the Quantizer sub-tab.

Procedure: 1. Prior to clicking the “Restore” button, open a new picture in the Manipulator.
2. Change some data values in the Manipulator to get the data input into the corresponding original text field.
3. Try clicking the “Restore” button to restore the originally loaded data into the corresponding tables.

Expected Result: The corresponding table field should be restored to the original value loaded from the image file.

Comments: Most images won’t have 4 Quantizer and/or 8 Huffman tables, so to test the unused fields, simply hit the “Restore” button, the original table will be restored to the corresponding field.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.3. General Tests

This section of the test plan is to test all of the other functions not covered by nay other section here. Each of the test in this section reflects some piece of the manipulator that has not previously been tested by any other section in the document.

3.2.3.1. TextBox Control Test

This section of the test plan is to test all of the TextBox controls found within the Manipulator. Each of the tests are described in their following section.

3.2.3.1.1. Changeable TextBox Control Tests

Purpose: To test the all “TextBox” controls in the Manipulator to make sure they are working properly.

Procedure: 1. Open a new picture in the Manipulator.

2. For each TextBox control that is not "grayed out," change some data values. If no data currently exists in the field, then just try adding some text into the control.

Expected Result: The data should be entered into the proper TextBox control, if the control is not "grayed out." If you encounter a control where this doesn't work, please write down the name of each one.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.3.1.2. Non-Changeable TextBox Control Tests

Purpose: To test the all non-changeable "TextBox" controls in the Manipulator to make sure they are working properly.

Procedure: 1. Open a new picture in the Manipulator.
2. For each TextBox control that is "grayed out," try to change some data values. If no data currently exists in the field, then just try adding some text into the control.

Expected Result: The data should NOT be entered into the proper TextBox control. If you encounter a control where this doesn't work, please write down the name of each one.

Comments: The non-changeable fields aren't as important as the changeable ones, but they should still all be checked. This will ensure that the original data won't be destroyed, so that the user can restore it if needed.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.3.1.3. Generating New JPEG Image Tests

Purpose: To test to make sure that the new image being created includes all of the values currently stored in the Manipulator and only those values.

Procedure:
1. Open a new picture in the Manipulator.
2. Try changing a bunch of different values for the picture.
3. Generate the new picture.
4. Use the converter to convert the original image and the newly generated image to an ASCII file and compare all of the data values.

Expected Result: The only data that should be changed in the newly generated image file from the original file is the data that was updated. Also, this updated data should be reflected in the new file as well.

Comments: The Converter should be sufficient to do this, but you may also want to run the newly generated picture through an image viewer (if the new image itself is viewable). You should use the Design document to evaluate whether or not the format of this file is correct.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.3.1.4. Generating New SEP Project File Tests

Purpose: To test to make sure that the new SEP file being created includes all of the values currently stored in the Manipulator and only those values.

Procedure:
1. Open a new picture in the Manipulator.
2. Try changing a bunch of different values for the picture.
3. Generated the new JPEG picture.
4. Add some project notes into the Project Notes text field.
5. Then save the SEP project file.
6. Then open the SEP file in some text processor, like NotePad or Word. You should be able to see all of the values saved in this file.

Expected Result: All of the changed file information should be stored in this file. Also the path and file name of both the original JPEG image and the changed JPEG image should be shown here as well. You should use the Design document to evaluate whether or not the format of this file is correct.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.3. Web Site Test

This section of the test plan document outlines the series of tests created to test all the functionality of the ISE Website. The web site test will be broken into the following categories:

1. **The Menu Frame Page**
2. **The Main Frame Pages**

The tests and results for both of these categories are compiled in the following sections of this document.

3.3.1. The Menu Frame Page

This section of the Test plan outlines tests done on the page Button.html, which appears in the Menu frame.

3.3.1.1. The Home Button

Purpose: This test is to verify that the Home button links to the correct page.

Procedure: Click the Home button on the Menu.

Expected Result: The page Home.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.2. The Project Proposal Button

Purpose: This test is to verify that the Project Proposal button links to the correct page.

Procedure: Click the Project Proposal button on the Menu.

Expected Result: The document ProjectProposal.pdf should open in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.3. The Documentation Button

Purpose: This test is to verify that the Documentation button links to the correct page.

Procedure: Click the Documentation button on the Menu.

Expected Result: The page DocumentIndex.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.4. The Project Sponsor Button

Purpose: This test is to verify that the Project Sponsor button links to the correct page.

Procedure: Click the Project Sponsor button on the Menu.

Expected Result: The page Sponsor.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.5. The Team Info Button

Purpose: This test is to verify that the Team Info button links to the correct page.

Procedure: Click the Team Info button on the Menu.

Expected Result: The page Team_ISE.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.6. The Download Button

Purpose: This test is to verify that the Download button links to the correct page.

Procedure: Click the Download button on the Menu.

Expected Result: The page Download.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.7. The Links Button

Purpose: This test is to verify that the Links button links to the correct page.

Procedure: Click the Links button on the Menu.

Expected Result: The page Links.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.8. The Message Board Button

Purpose: This test is to verify that the Message Board button links to the correct page.

Procedure: Click the Message Board button on the Menu.

Expected Result: The page index.php should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2. The Main Frame Pages

This section outlines the tests done on the various pages displayed in the Main frame.

3.3.2.1. DocumentIndex.html Requirements Button

Purpose:	This test is to verify that the Requirements button on the DocumentIndex.html page functions correctly.
Procedure:	Click the Requirements button on the page.
Expected Result:	A .pdf reader should open ISEFinalRequirements.pdf file in the Main frame.
Comments:	None.
Date:	March 6, 2004
Tester:	Andrew Pouzeshi
Outcome:	Pass

3.3.2.2. DocumentIndex.html Prototype Plan Button

Purpose:	This test is to verify that the Prototype Plan button on the DocumentIndex.html page functions correctly.
Procedure:	Click the Prototype Plan button on the page.
Expected Result:	A .pdf reader should open ISEPrototypePlan.pdf file in the Main frame.
Comments:	None.
Date:	March 6, 2004
Tester:	Andrew Pouzeshi
Outcome:	Pass

3.3.2.3. DocumentIndex.html Sys Arch Design Button

Purpose:	This test is to verify that the Sys Arch Design button on the DocumentIndex.html page functions correctly.

Procedure: Click the Sys Arch Design button on the page.

Expected Result: A .pdf reader should open ISESystemArchitectureDesign.pdf file in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.4. DocumentIndex.html Design Document Button

Purpose: This test is to verify that the Design Document button on the DocumentIndex.html page functions correctly.

Procedure: Click the Design Document button on the page.

Expected Result: A .pdf reader should open DesignSpecFinal.pdf file in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.5. Sponsor.html Project Sponsor Button

Purpose: This test is to verify that the Project Sponsor button on the Sponsor.html page functions correctly.

Procedure: Click the Project Sponsor button on the page.

Expected Result: The page located at http://www.cs.colorado.edu/people/tom_lookabaugh.html should be displayed in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.6. Download.html Production Code Button

Purpose: This test is to verify that the Production Code button on the Download.html page functions correctly.

Procedure: Click the Production Code button on the page.

Expected Result: The browser should prompt a window asking the user where they would like to download the zip file code.zip.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.7. Download.html Manipulator Button

Purpose: This test is to verify that the Manipulator button on the Download.html page functions correctly.

Procedure: Click the Manipulator button on the page.

Expected Result: The browser should prompt a window asking the user where they would like to download the zip file manipulator.zip.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.8. Download.html .NET Framework Button

Purpose: This test is to verify that the .NET Framework button on the Download.html page functions correctly.

Procedure: Click the .NET Framework button on the page.

Expected Result: The browser should prompt a window asking the user where they would like to download the file dotnetfx.exe.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.9. Download.html Alpha Test Button

Purpose: This test is to verify that the .NET Framework button on the Download.html page functions correctly.

Procedure: Click the .NET Framework button on the page.

Expected Result: The browser should prompt a window asking the user where they would like to download the file dotnetfx.exe.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.10. Links.html www.ijg.org/ Button

Purpose: This test is to verify that the www.ijg.org/ button on the Links.html page functions correctly.

Procedure: Click the www.ijg.org/ button on the page.

Expected Result: The page located at http://www.ijg.org should be displayed in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.11. Links.html rijndael algo Button

Purpose: This test is to verify that the rijndael algo button on the Links.html page functions correctly.

Procedure: Click the rijndael button on the page.

Expected Result: The page located at http://www.esat.kuleuven.ac.be/~rijmen/rijndael/ should be displayed in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.12. Links.html Project Sponsor Button

Purpose: This test is to verify that the Project Sponsor button on the Links.html page functions correctly.

Procedure: Click the Project Sponsor button on the page.

Expected Result: The page located at http://www.cs.colorado.edu/people/tom_lookabaugh.html should be displayed in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

4. SUMMARY

This document gives a detailed test plan for the ISE Production Code, Manipulator, and the ISE web site. These tests should be sufficient to prove that the Production Code, Manipulator, and web site are functioning correctly. All of the results uncovered by the team members during the alpha testing process have been listed here as well.

5. RELATED READINGS

[Chang and Li 96]

Chang, H. and Li, X. *On the Application of Image Decomposition to Image Compression and Encryption*. 1996.

Describes image degradation based on compression and encryption.

[Chang and Li 2000]

Chang, H. and Li, X. *Partial Encryption of Compressed Images and Videos*. 2000.

Describes a partial encryption scheme used on compressed multimedia files.

[Droogenbroek and Benedett 2002]

Droogenbroek, M. and Benedett, R. *Techniques for Selective Encryption of Uncompressed and Compressed Images*. 2002.

[Kailasanathan and Naini 2003]

Kailasanathan, C. and Naini, R. *Compression Performance of JPEG Encryption Scheme*. 2003.

Describes compression performance of JPEG encryption.

[Daigaku and Griffith and Jarchow and Kadhim and Pouzeshi]

Daigaku, S., Griffith, G., Jarchow, J., Kadhim, J. and Pouzeshi A. *Requirement Specification.* 2003.

Describes the requirement for Team ISE and for the ISE project.

[Daigaku and Griffith and Jarchow and Kadhim and Pouzeshi]

Daigaku, S., Griffith, G., Jarchow, J., Kadhim, J. and Pouzeshi A. *System Architecture.* 2003.

Describes the high-level system architecture for the ISE project.

[Li and Knipe and Cheng 97]

Li, X., Knipe, J. and Cheng, H. *Image Compression and Encryption Using Tree Structures*. 1997.

Describes compression methods that utilize tree structures.

[Lookabaugh and Sicker and Keaton and Guoand and Vedula 2003]
Lookabaugh, T., Sicker, D., Keaton, D., Guoand, W. and Vedula, I. *Security Analysis of Selectively Encrypted MPEG-e Streams*. 2003.

Description of the methods and results of applying selective encryption to MPEG-2 streams.

[Miano 99]
Miano, J. *Compressed Image File Formats*. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

Provides a description of the JPEG file format.

[Norcen and Uhl 2003]
Norcen, R. and Uhl, A. *Selective Encryption of the JPEG2000 Bitstream*. 2003.

Describes a selective encryption scheme on JPEG2000 files.

[Pennebaker and Mitchell 93]
Pennebaker, W. and Mitchell J. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, New York, 1993,

Provides a thorough description of the JPEG file format and its components.

[Podesser and Schmidt and Uhl 2002]
Podesser, M., Schmidt, H. and Uhl, A. *Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments*. 2002.

Describes Bitplane Encryption.

[Seo and Kim and Yoo and Dey and Agrawal 2003]
Seo, Y., Kim, D., Yoo, J., Dey, S., Agrawal, A. *Wavelet Domain Imag Encryption by Subband Selection and Data Bit Selection*. 2003.

Describes Wavelet Domain and Data Bit encryption methods.

ISE Class Man Pages

Team ISE

Image Selective Encryption

Table of Contents

NAME
ise::ise()

SYNOPSIS
#include <ise.h>
ise::ise(char* key, char* input_file_name, char* output_file_name);

DESCRIPTION
ise::ise()
Only classes that extend the ise class use this constructor.
An ise object is constructed with the data necessary to encrypt or decrypt a file.
This constructor only requires that the key be provided. The input_file_name and output_file_name arguments are optional and will be set to a default value of NULL.
For more information about the ise class, please visit http://128.138.75.184.

PRE-CONDITIONS
key must be a pointer to a character string.

POST-CONDITIONS
An ise object is created containing the specified data members.

PARAMETERS
key is a pointer to the encryption/decryption key.
input_file_name is the name and path of the input file to be encrypted or decrypted.
output_file_name is the file name and path for the output file generated by encryption or decryption.

EXAMPLE USEAGE
Only classes that extend the ise class use this constructor.

SEE ALSO
jpeg_ise::jpeg_ise()

TEAM ISE Last Change: 15 April 2004

ISE Production Code Functions
jpeg_ise::jpeg_ise(3)

NAME
jpeg_ise::jpeg_ise()

SYNOPSIS
#include <ise.h>
jpeg_ise::jpeg_ise(char* key, char* input_file_name, char* output_file_name)

DESCRIPTION
jpeg_ise::jpeg_ise()
An ise object is constructed with the data necessary to encrypt or decrypt a file.
This overloaded constructor only requires that key be provided.
The input_file_name and output_file_name arguments are optional and will be set to a default value of NULL.
For more information on the jpeg_ise and ise classes, please visit http://128.138.75.184.

PRE-CONDITIONS
key must be a pointer to a character string.

POST-CONDITIONS
A jpeg_ise object is created containing the specified data members.

PARAMETERS
key is a pointer to the encryption key.
input_file_name is the name and path of the input file to be encrypted or decrypted.
output_file_name is the file name and path for the output file generated by encryption or decryption.

EXAMPLE USEAGE
// constructor with only the key
jpeg_ise MyJpegIseObj (MyKey);

// constructor with all arguments for encryption
jpeg_ise MyJpegIseObj (MyKey, MyJpeg, MyIse);

// constructor with all arguments for decryption
jpeg_ise MyJpegIseObj (MyKey, MyIse, MyJpeg);

SEE ALSO
ise::ise()

TEAM ISE Last Change: 15 April 2004

ISE Production Code Functions
jpeg_ise::encrypt_file(0)

NAME
jpeg_ise::encrypt_file()

SYNOPSIS
#include <ise.h>
int jpeg_ise::encrypt_file()

DESCRIPTION
jpeg_ise::encrypt_file()
The encrypt_file() method will take a standard baseline compression jpeg file and selectively encrypt the Huffman Table frames found within the file.
If the output file already exists, the existing file will be overwritten, otherwise, a new encrypted file will be created for the selectively encrypted jpeg image.

PRE-CONDITIONS
The input_file_name and key ise data memberds must be set using either the overloaded constructor or the set_input_file_name(char* name) and set_key(char* key) functions prior to calling this method.
This function requires that the input and output file pointers be at the head of the file.

POST-CONDITIONS
An encrypted file will be created with the name and path specified by the output_file_name data member.
If this data member is NULL, then a default file name will be created based upon the input_file_name data member.

RETURN VALUES
An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate could not open input file name.
2 will indicate could not create ise file name.
3 will indicate could not open ise file.

EXAMPLE USEAGE
```
// MyJpegIseObj must be of type jpeg_ise
// encrypt jpeg file to ise file
MyJpegIseObj.encrypt_file();
```

SEE ALSO
ise::ise(), jpeg_ise::jpeg_ise(), ise::set_input_file_name(), ise::set_output_file_name(), ise::set_key(), jpeg_ise::decrypt_file()

TEAM ISE Last Change: 15 April 2004

ISE Production Code Functions
jpeg_ise::decrypt_file(0)

NAME
jpeg_ise::decrypt_file()

SYNOPSIS
#include <ise.h>
int jpeg_ise::decrypt_file()

DESCRIPTION
jpeg_ise::decrypt_file()
The decrypt_file method will take a jpeg_ise file and selectively decrypt the Huffman Table frames found within the file.
If the output file already exists, the existing file will be overwritten. Otherwise, a new file will be created for the selectively decrypted jpeg image.

PRE-CONDITIONS
The input_file_name and key ise data members must be set using either the jpeg_ise() overloaded constructor or the set_input_file_name(char* name) and set_key(char* key) functions prior to calling this method.
This code requires that the input and output file pointers be at the head of the file.

POST-CONDITIONS
A decrypted file will be created with the name and path specified by the output_file_name data member.
If this data member is NULL, then a default file name will be created based upon the input_file_name data member.

RETURN VALUES
An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate input file is not a jpeg ise file.
2 will indicate could not open ise file.
3 will indicate could not create output jpeg file.
4 will indicate could not open output jpeg file.

EXAMPLE USEAGE
```
// MyJpegIseObj must be of type jpeg_ise
// decrypt ise file to jpeg file
MyJpegIseObj.decrypt_file();
```

SEE ALSO
ise::ise(), jpeg_ise::jpeg_ise(), ise::set_input_file_name(), ise::set_output_file_name(), ise::set_key(), jpeg_ise::encrypt_file()

TEAM ISE Last Change: 15 April 2004

ISE Production Code Functions
ise::set_key(1)

NAME
ise::set_key()

SYNOPSIS
#include <ise.h>
int ise::set_key(char* name)

DESCRIPTION
ise::set_key()
The method will use the specified name to create a valid key to be used by the ise encryption or decryption methods.

PRE-CONDITIONS
name must be a pointer to a character string.

POST-CONDITIONS
The key data member will be set using the new string specified.
Any previous information in the key will be lost.

PARAMETERS
name is a pointer to a character string containing the key information for either encryption or decryption.

RETURN VALUES
An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate an invalid key.

EXAMPLE USEAGE
```
// create key for encryption/decryption
char MyKey[] = "EnterKeyHere";

// MyIseObj must be of type ise or an inheriting class
// set the key
 MyIseObj.set_key(MyKey);
```

SEE ALSO
ise::ise(), jpeg_ise::jpeg_ise(), ise::set_input_file_name(), ise::set_output_file_name()

TEAM ISE Last Change: 15 April 2004

ISE Production Code Functions
ise::set_input_file_name(1)

NAME
ise::set_input_file_name()

SYNOPSIS
#include <ise.h>
int ise::set_input_file_name(char* name)

DESCRIPTION
ise::set_input_file_name()
This method is used to set the input_file_name data member for an ise object.
The method must be called prior to the encryption or decryption methods if the input_file_name was not specified in the constructor.

PRE-CONDITIONS
name must be a pointer to a valid jpeg or ise file type.

POST-CONDITIONS
The input_file_name data member will be set using the new string specified.
Any previous data in input_file_name will be lost.

PARAMETERS
name is a pointer to a character string containing the input_file_name, specifying the input file to encryption or decryption.

RETURN VALUES
An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate an invalid input file name.

EXAMPLE USEAGE
```
// MyJpegIseIbj must be of type jpeg_ise
// set a jpeg input file for encryption
MyJpegIseObj.set_input_file_name(MyJpeg);

// set an ise input file for decryption
MyJpegIseObj.set_input_file_name(MyISE);
```

SEE ALSO
ise::ise(), jpeg_ise::jpeg_ise(), ise::set_key(), ise::set_output_file_name()

TEAM ISE Last Change: 15 April 2004

NAME
ise::set_output_file_name()

SYNOPSIS
#include <ise.h>
int ise::set_output_file_name(char* name)

DESCRIPTION
ise::set_output_file_name()
This method is used to set the output_file_name ise data member.

PRE-CONDITIONS
name must be a pointer to a valid jpeg or ise file type.

POST-CONDITIONS
The output_file_name data member will be set using the new string specified.
Any previous data in output_file_name will be lost.

PARAMETERS
name is a pointer to a character string containing the output_file_name, specifying the output file to encryption or decryption.

RETURN VALUES
An integer is returned indicating a success or failure.
A 0 will indicate a success.
A 1 will indicate an invalid output file name.

EXAMPLE USEAGE
// MyJpegIseObj must be of type jpeg_ise
// set a jpeg output file for decryption
MyJpegIseObj.set_output_file_name(MyJpeg);

// set an ise output file for encryption
MyJpegIseObj.set_output_file_name(MyISE);

SEE ALSO
ise::ise(), jpeg_ise::jpeg_ise(), ise::set_key(), ise::set_input_file_name()

TEAM ISE Last Change: 15 April 2004

NAME
ise::get_input_file_name()

SYNOPSIS
#include <ise.h>
int ise::get_input_file_name()

DESCRIPTION
char* ise::get_input_file_name()
This is the accessor method for the input_file_name ise data member.

RETURN VALUES
The method will return the input_file_name data member as a character string.
If the input_file_name is not set, the method will return NULL.

EXAMPLE USEAGE
// MyIseObj must be of type ise or an inheriting class
fileName = MyIseObj.get_input_file_name();

SEE ALSO
ise::get_output_file_name(), ise::get_ise_file_type(), ise::set_input_file_name()

TEAM ISE Last Change: 15 April 2004

NAME
ise::get_output_file_name()

SYNOPSIS
#include <ise.h>
char* ise::get_output_file_name()

DESCRIPTION
ise::get_output_file_name()
This is the accessor method for the output_file_name ise data member.

RETURN VALUES
The method will return the output_file_name data member as a character string.
If the output_file_name is not set, the method will return NULL.

EXAMPLE USEAGE
// MyIseObj must be of type ise or an inheriting class.
fileName = MyIseObj.get_output_file_name();

SEE ALSO
ise::get_input_file_name(), ise::get_ise_file_type(), ise::set_input_file_name()

TEAM ISE Last Change: 15 April 2004

ISE Class Reference

Team ISE

Image Selective Encryption

Table of Contents

NAME

ise::ise()

SYNOPSIS

#include <ise.h>

ise::ise(char* **key**, char* **input_file_name**, char* **output_file_name**);

DESCRIPTION

ise::ise()

Only classes that extend the **ise** class use this constructor.

An **ise** object is constructed with the data necessary to encrypt or decrypt a file. This constructor only requires that the **key** be provided. The **input_file_name** and **output_file_name** arguments are optional and will be set to a default value of NULL.

For more information about the ise class, please visit http://128.138.75.184.

PRE-CONDITIONS

key must be a pointer to a character string.

POST-CONDITIONS

An **ise** object is created containing the specified data members.

PARAMETERS

key is a pointer to the encryption/decryption key.

input_file_name is the name and path of the input file to be encrypted or decrypted.

output_file_name is the file name and path for the output file generated by encryption or decryption.

EXAMPLE USEAGE

Only classes that extend the **ise** class use this constructor.

SEE ALSO

jpeg_ise::jpeg_ise()

TEAM ISE Last Change: 15 April 2004

NAME

jpeg_ise::jpeg_ise()

SYNOPSIS

#include <ise.h>

jpeg_ise::jpeg_ise(char* **key**, char* **input_file_name**, char* **output_file_name**)

DESCRIPTION

jpeg_ise::jpeg_ise()

An ise object is constructed with the data necessary to encrypt or decrypt a file. This overloaded constructor only requires that **key** be provided.

The **input_file_name** and **output_file_name** arguments are optional and will be set to a default value of NULL.

For more information on the jpeg_ise and ise classes, please visit http://128.138.75.184.

PRE-CONDITIONS

key must be a pointer to a character string.

POST-CONDITIONS

A jpeg_ise object is created containing the specified data members.

PARAMETERS

key is a pointer to the encryption key.

input_file_name is the name and path of the input file to be encrypted or decrypted.

output_file_name is the file name and path for the output file generated by encryption or decryption.

EXAMPLE USEAGE

```
// constructor with only the key
jpeg_ise MyJpegIseObj (MyKey);

// constructor with all arguments for encryption
jpeg_ise MyJpegIseObj (MyKey, MyJpeg, MyIse);

// constructor with all arguments for decryption
jpeg_ise MyJpegIseObj (MyKey, MyIse, MyJpeg);
```

SEE ALSO

ise::ise()

TEAM ISE Last Change: 15 April 2004

NAME

jpeg_ise::encrypt_file()

SYNOPSIS

#include <ise.h>
int jpeg_ise::encrypt_file()

DESCRIPTION

jpeg_ise::encrypt_file()
The encrypt_file() method will take a standard baseline compression jpeg file and selectively encrypt the Huffman Table frames found within the file.
If the output file already exists, the existing file will be overwritten, otherwise, a new encrypted file will be created for the selectively encrypted jpeg image.

PRE-CONDITIONS

The **input_file_name** and **key** ise data memberds must be set using either the overloaded constructor or the set_input_file_name(char* **name**) and set_key(char* **key**) functions prior to calling this method.
This function requires that the input and output file pointers be at the head of the file.

POST-CONDITIONS

An encrypted file will be created with the name and path specified by the **output_file_name** data member.
If this data member is NULL, then a default file name will be created based upon the **input_file_name** data member.

RETURN VALUES

An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate could not open input file name.
2 will indicate could not create ise file name.
3 will indicate could not open ise file.

EXAMPLE USEAGE

```
// MyJpegIseObj must be of type jpeg_ise
// encrypt jpeg file to ise file
MyJpegIseObj.encrypt_file();
```

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), ise::set_input_file_name(), ise::set_output_file_name(), ise::set_key(), jpeg_ise::decrypt_file()

TEAM ISE Last Change: 15 April 2004

NAME

jpeg_ise::decrypt_file()

SYNOPSIS

#include <ise.h>
int jpeg_ise::decrypt_file()

DESCRIPTION

jpeg_ise::decrypt_file()
The decrypt_file method will take a jpeg_ise file and selectively decrypt the Huffman Table frames found within the file.
If the output file already exists, the existing file will be overwritten. Otherwise, a new file will be created for the selectively decrypted jpeg image.

PRE-CONDITIONS

The **input_file_name** and **key** ise data members must be set using either the jpeg_ise() overloaded constructor or the set_input_file_name(char* **name**) and set_key(char* **key**) functions prior to calling this method.
This code requires that the input and output file pointers be at the head of the file.

POST-CONDITIONS

A decrypted file will be created with the name and path specified by the **output_file_name** data member.
If this data member is NULL, then a default file name will be created based upon the **input_file_name** data member.

RETURN VALUES

An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate input file is not a jpeg ise file.
2 will indicate could not open ise file.
3 will indicate could not create output jpeg file.
4 will indicate could not open output jpeg file.

EXAMPLE USEAGE

```
// MyJpegIseObj must be of type jpeg_ise
// decrypt ise file to jpeg file
MyJpegIseObj.decrypt_file();
```

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), ise::set_input_file_name(), ise::set_output_file_name(), ise::set_key(), jpeg_ise::encrypt_file()

TEAM ISE Last Change: 15 April 2004

NAME

ise::set_key()

SYNOPSIS

#include <ise.h>
int ise::set_key(char* **name**)

DESCRIPTION

ise::set_key()
The method will use the specified **name** to create a valid key to be used by the ise encryption or decryption methods.

PRE-CONDITIONS

name must be a pointer to a character string.

POST-CONDITIONS

The **key** data member will be set using the new string specified.
Any previous information in the **key** will be lost.

PARAMETERS

name is a pointer to a character string containing the key information for either encryption or decryption.

RETURN VALUES

An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate an invalid key.

EXAMPLE USEAGE

```
// create key for encryption/decryption
char MyKey[] = "EnterKeyHere";

// MyIseObj must be of type ise or an inheriting class
// set the key
 MyIseObj.set_key(MyKey);
```

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), ise::set_input_file_name(), ise::set_output_file_name()

TEAM ISE Last Change: 15 April 2004

NAME

ise::set_input_file_name()

SYNOPSIS

#include <ise.h>
int ise::set_input_file_name(char* **name**)

DESCRIPTION

ise::set_input_file_name()
This method is used to set the **input_file_name** data member for an ise object. The method must be called prior to the encryption or decryption methods if the **input_file_name** was not specified in the constructor.

PRE-CONDITIONS

name must be a pointer to a valid jpeg or ise file type.

POST-CONDITIONS

The **input_file_name** data member will be set using the new string specified. Any previous data in **input_file_name** will be lost.

PARAMETERS

name is a pointer to a character string containing the **input_file_name**, specifying the input file to encryption or decryption.

RETURN VALUES

An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate an invalid input file name.

EXAMPLE USEAGE

```
// MyJpegIseIbj must be of type jpeg_ise
// set a jpeg input file for encryption
MyJpegIseObj.set_input_file_name(MyJpeg);

// set an ise input file for decryption
MyJpegIseObj.set_input_file_name(MyISE);
```

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), ise::set_key(), ise::set_output_file_name()

TEAM ISE Last Change: 15 April 2004

NAME

ise::set_output_file_name()

SYNOPSIS

#include <ise.h>
int ise::set_output_file_name(char* **name**)

DESCRIPTION

ise::set_output_file_name()
This method is used to set the **output_file_name** ise data member.

PRE-CONDITIONS

name must be a pointer to a valid jpeg or ise file type.

POST-CONDITIONS

The **output_file_name** data member will be set using the new string specified.
Any previous data in **output_file_name** will be lost.

PARAMETERS

name is a pointer to a character string containing the **output_file_name**, specifying the output file to encryption or decryption.

RETURN VALUES

An integer is returned indicating a success or failure.
A 0 will indicate a success.
A 1 will indicate an invalid output file name.

EXAMPLE USEAGE

```
// MyJpegIseObj must be of type jpeg_ise
// set a jpeg output file for decryption
MyJpegIseObj.set_output_file_name(MyJpeg);

// set an ise output file for encryption
MyJpegIseObj.set_output_file_name(MyISE);
```

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), ise::set_key(), ise::set_input_file_name()

TEAM ISE Last Change: 15 April 2004

NAME

ise::get_key()

SYNOPSIS

#include <ise.h>
char * ise::get_key()

DESCRIPTION

ise::get_key()
This method will return a char pointer for the **key** string.
This is an ise class protected function, and is only called from within inheriting classes.

RETURN VALUES

This method will return a char pointer for the **key** string.

EXAMPLE USEAGE

This function is only called by inheriting classes.

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), jpeg_ise::encrypt_file(), jpeg_ise::decrypt_file(), ise::get_key()

TEAM ISE Last Change: 18 April 2004

NAME

ise::get_input_file_name()

SYNOPSIS

#include <ise.h>
int ise::get_input_file_name()

DESCRIPTION

char* ise::get_input_file_name()
This is the accessor method for the **input_file_name** ise data member.

RETURN VALUES

The method will return the **input_file_name** data member as a character string.
If the **input_file_name** is not set, the method will return NULL.

EXAMPLE USEAGE

```
// MyIseObj must be of type ise or an inheriting class
fileName = MyIseObj.get_input_file_name();
```

SEE ALSO

ise::get_output_file_name(), ise::get_ise_file_type(), ise::set_input_file_name()

TEAM ISE Last Change: 15 April 2004

NAME

ise::get_output_file_name()

SYNOPSIS

#include <ise.h>
char* ise::get_output_file_name()

DESCRIPTION

ise::get_output_file_name()
This is the accessor method for the **output_file_name** ise data member.

RETURN VALUES

The method will return the **output_file_name** data member as a character string.
If the **output_file_name** is not set, the method will return NULL.

EXAMPLE USEAGE

```
// MyIseObj must be of type ise or an inheriting class.
fileName = MyIseObj.get_output_file_name();
```

SEE ALSO

ise::get_input_file_name(), ise::get_ise_file_type(), ise::set_input_file_name()

TEAM ISE Last Change: 15 April 2004

NAME

ise::make_ise_file_name()

SYNOPSIS

#include <ise.h>
int ise::make_ise_file_name()

DESCRIPTION

ise::make_ise_file_name()
The file name and path created will be the same as the string pointed to by the **input_file_name** ise data member, except that the extension of the file will be changed to .ise.
If this file already exists, then a 0 will be added on to the end of the file name, just before the extension.
If this file already exists, the function will keep incrementing this number and checking, until the new file name does not previously exist.
This is an ise class private function, and is only called within the ise class.

PRE-CONDITIONS

The user of the class has previously set the **input_file_name** data member.

POST-CONDITIONS

The **output_file_name** data member points to a string with a file name and file path, based upon the string pointed to by the **input_file_name** data member.

RETURN VALUES

An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate a failure.

EXAMPLE USEAGE

This function is called privately by other ise functions.

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), ise::set_input_file_name()

TEAM ISE Last Change: 15 April 2004

NAME

ise::make_output_file_name()

SYNOPSIS

#include <ise.h>
int ise::make_output_file_name();

DESCRIPTION

ise::make_output_file_name()
The file name and path created will be the same as the string pointed to by the **input_file_name** ise data member, except that the extension of the file will be changed to .jpg.
If this file already exists, then a 0 will be added on to the end of the file name, just before the extension.
If this file already exists, the function will keep incrementing this number and checking, until the new file name does not previously exist.
This is an ise class private function and is only called from within the ise class.

PRE-CONDITIONS

The user of the class has previously set the **input_file_name** data member.

POST-CONDITIONS

The **output_file_name** data member points to a string with a file name and file path, based upon the string pointed to by the **input_file_name** data member.

RETURN VALUES

An integer is returned indicating a success or failure.
0 will indicate a success.
1 will indicate a failure.

EXAMPLE USEAGE

This function is called privately by other ise functions.

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), ise::set_input_file_name()

TEAM ISE Last Change: 15 April 2004

NAME

ise::get_ise_file_type()

SYNOPSIS

#include <ise.h>
int ise::get_ise_file_type()

DESCRIPTION

ise::get_ise_file_type()
This method will return an integer corresponding to the original file type of an encrypted ise file.

RETURN VALUES

The function will return an integer indicating the type of the original file from which the specified ise file was created.
0 will indicate an unknown or unimplemented file type.
1 will indicate a jpeg file.
The return values may be extended to accommodate other file types.

EXAMPLE USEAGE

```
// MyIseObj must be of type ise or an inheriting class.
fileType = MyIseObj.get_ise_file_type();
```

SEE ALSO

ise::ise(), jpeg_ise::jpeg_ise(), jpeg_ise::encrypt_file(), jpeg_ise::decrypt_file()

TEAM ISE Last Change: 15 April 2004

```
///--------------------------------------------------------------------------
///
///  File Name:         Main.cpp
///
///  File Description:
///
///  This file is designed as an example program using the ISE class
///  functionality.  A user can modify this program by uncommenting the extra
///  code and making this file.
///
///   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.
///
///--------------------------------------------------------------------------

#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <string>
#include "rijndael-api-fst.h"
#include "ise.h"
```

```
using namespace std;


int main(int argc, char * argv[])
{
  // key used for encryption and decryption
  char MyKey[] = "EnterKeyHere";

  // original jpeg file to be encrypted
  char * MyJpeg = "c:/ralphie.jpg";

  // name for ise file created during encryption
  char * MyIse = "c:/gumble.ise";

  // name for new jpeg after decryption
  char * finalJpeg = "c:/newralphie.jpg";

  int err;

  // call the constructor with only the key
  jpeg_ise test(MyKey);

  // call the constructor with all arguments
  //jpeg_ise test(dumKey, MyJpeg, MyIse);


  /************** Encrypt ********************************************/
  // set the key
  //err = test.set_key(MyKey);
  //if (err != 0) cout << "set_key() failed!\n";

  // set the input jpeg file
  err = test.set_input_file_name(MyJpeg);
  if (err != 0) cout << "set_input_file_name() failed!\n";

  // set the output ise file
  err = test.set_output_file_name(MyIse);
  if(err != 0) cout << "set_output_file_name() failed!\n";

  // test get input/output file names functions
  cout << "Input file name:  " << test.get_input_file_name() << "\n";
  cout << "Output file name: " << test.get_output_file_name() << "\n";

  // encrypt the file
  err = test.encrypt_file();
```

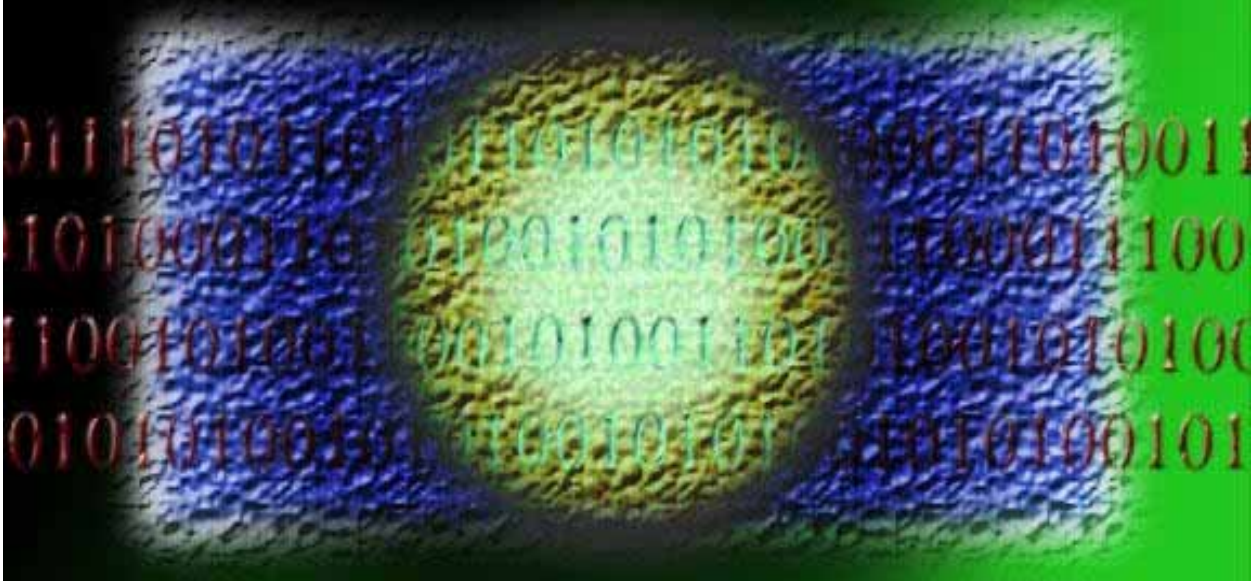
```
  if (err != 0) cout << "Encryption failed!\n";
  else cout << "File encryption successful!\n";

  /************ Decrypt ***********************************************/
  // set the input ise file
  err = test.set_input_file_name(MyIse);
  if(err != 0) cout << "set_input_file_name() failed!\n";

  // set the output jpeg file
  err = test.set_output_file_name(finalJpeg);
  if(err != 0) cout << "set_output_file_name() failed!\n";

  // test get input/output file names functions
  cout << "Input file name:  " << test.get_input_file_name() << "\n";
  cout << "Output file name: " << test.get_output_file_name() << "\n";

  // decrypt the file
  err = test.decrypt_file();
  if (err != 0) cout << "Decryption failed!\n";
  else cout << "File decryption successful!\n";

  return 0;
}
```

Manipulator Tutorial

Team ISE

Image Selective Encryption

Manipulator Tutorial Manual



Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Table of Contents

Introduction:

The ISE JPEG Manipulator is an application designed to allow the user to examine, manipulate and create images from the data of pre-existing JPEG images. This document is a short tutorial to provide users with step-by-step instruction for some the Manipulator's more common operations. This document uses the "splash.jpg" image included with the Manipulator in the main program directory. You can use this image to follow along with the tutorial. Also, for a complete listing of all of the specific functionality of the ISE JPEG Manipulator, be sure to refer to the Manipulator User Manual.

1. Installing the ISE JPEG Manipulator:

The Manipulator comes prepackaged with a full installation script that performs all the necessary functions to properly install the Manipulator with a minimal degree of user effort. If a previous version of the Manipulator was installed prior to this installation, be sure to completely remove the previous version before preceding with this installation, otherwise this installation will not complete properly. Uninstalling the Manipulator is covered under section 1.4 of the user manual or section 2 of this document.

The Manipulator has several system requirements needed as a minimum to properly install and run the Manipulator. These requirements are as follows:

1. Microsoft Windows NT/2000/XP Operating System.
2. Microsoft .NET framework version 1.1.
3. A monitor, mouse and keyboard for the host computer.
4. 100 MB of free Hard Disk space.
5. 300 MHz or faster CPU.
6. 64 MB of RAM.

If your system meets all minimum system requirements and there is no previous version of the Manipulator installed, you are now ready to begin the installation process. To install the ISE JPEG Manipulator, complete the following steps:

1. If the ISE JPEG Manipulator has not been previously downloaded, be sure to download the installation package to the computer where you wish to install the program. Be sure to save the download some place that can be easily accessed, like your "Desktop." You can download the JPEG Manipulator from:

   http://128.138.75.184/code/ISEManipulator107.zip

2. Once the file has completed downloading, double-click on the file to begin the installation process.

3. Follow through on-screen instructions to successfully complete the installation. Team ISE recommends using all of the default settings for installation.

2. Uninstalling the ISE JPEG Manipulator:

The Manipulator installation package also includes an uninstall script to remove all of the application files, if the need arises. The uninstaller will remove all data copied and created during the installation process. Please note that images created by the user or any original JPEG pictures used will not be removed, unless they are saved within the ISE program folder (the folder created in the programs files during installation). To perform the uninstall process, complete the following steps:

1. Go to "Start" >> "Settings" >> "Control Panel"

2. Once you click on the "Control Panel," you should see the contents of the Control Panel folder. Double-click on the "Add/Remove Programs" icon.

3. Within the Add/Remove Programs utility, find the "JPEG Manipulator" entry and click on it to highlight it in blue and then click on the "Remove" button next to it.

4. Follow through the on-screen instructions to successfully complete removal of the application.

Please note that you must uninstall any previous versions of the ISE JPEG Manipulator before installing any updated versions.

3. Running the JPEG Manipulator Application:

Once the Manipulator has successfully completed installation, you should be able to instantiate the application without further delay. If the default installation was chosen, then the program can be invoked by going to:

Start Button >> Programs >> ISE >> JPEG >> JPEG Manipulator

Otherwise, if a different location for the program menu has been chosen, then go to the folder where the program was installed and then go to:

ISE >> JPEG >> JPEG Manipulator

Once the JPEG Manipulator icon is clicked, you will start the JPEG Manipulator application and can begin working with JPEG images. Although the application is designed to work specifically with the Baseline compression standard for JPEG images, the application should work with other JPEG formats as well.

Once the application has been opened, you should be at the main application window. Figure 1 is a picture of the main application window:

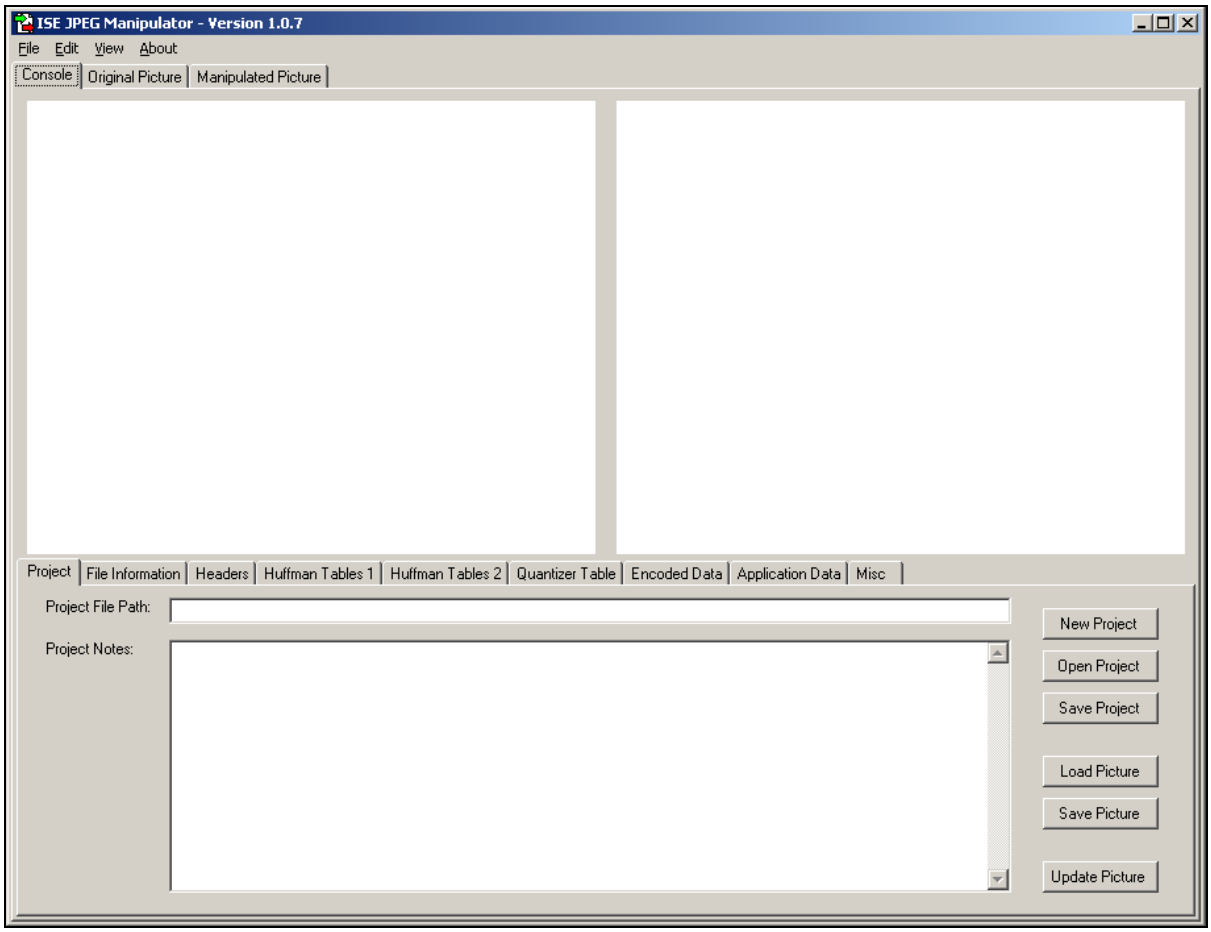


Figure 1: The Manipulator Main Window.

From this window you can perform almost any function within the application and this is the main window you will do work from. When the application is invoked, the "Console" tab (on the top-right of the window) will be the default tab selected and is where most of the work in the application is done. The "Console" tab has a number of sub-tabs that contain all of the data for the current project and/or images loaded within the application. Also, it is possible to view both the original and the manipulated images in a larger window by clicking on the "Original Picture" or "Manipulated Picture" tabs (respectively), located to the right of the "Console" tab. At the top of the window is a main menu to allow the user to perform common operations during use of the application.

Each of the sub-tabs found on the "Console" tab are related to the data found in a JPEG image or a Project file. The following is a brief description of each sub-tab's purpose:

1. "Project" sub-tab: Contains project file and project note data. Also contains buttons for performing common tasks within the manipulator.
2. "File Information" sub-tab: Contains original and manipulated picture file names and paths and any file comment data.
3. "Headers" sub-tab: Contains the data found in the SOF0 frame, if the image loaded is a Baseline standard compression.

4. “Huffman Tables 1 & 2” sub-tabs: Contains the compression data found in any compression frame markers ffc1 to ffcf in the JPEG image.
5. “Quantizer Table” sub-tab: Contains the data found in the DQT frames of the JPEG image.
6. “Encoded Data” sub-tab: Contains the SOS frame data and the first 20,000 bytes of the encoded data stream.
7. “Application Data” sub-tab: Contains the data for all of the APP frames (if included with the JPEG image).
8. “Misc” sub-tab: Contains any other frame data not included on any other tab.

The following sections outline the use of only some of the functionality here. Please refer to the Manipulator User Manual for a complete listing of this functionality.

4. Loading a JPEG image:

The best way to begin is to Load a JPEG image for editing. There are two ways to do this within the application: on the main menu or on the “Project” sub-tab located on the “Console” tab. The following are detailed, step-by-step instructions for loading images:

1. Click on the “File” menu option. This will cause the menu to become visible at the top of the window (shown in figure 2).

2. Then click on the “Load Picture” menu option (shown in figure 2). This will cause the “Open JPEG File” dialog box to appear (shown in figure 3).

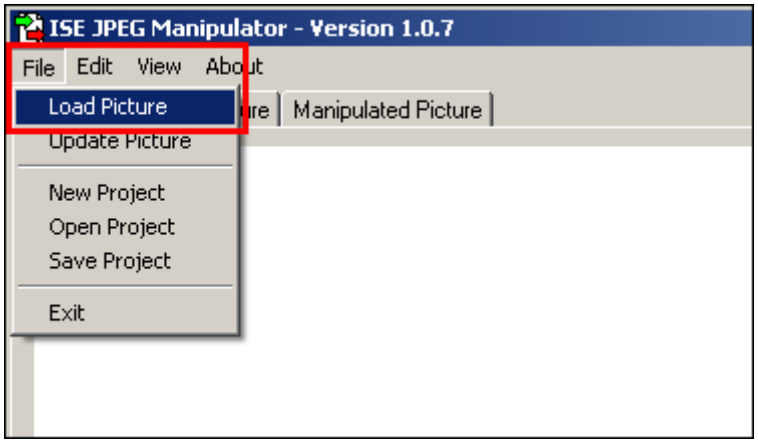


Figure 2: The “File” Menu (Load Picture Highlighted)

3. Then select an image by browsing to it and clicking on the desired image (shown in figure 3). Only images that have a JPEG extension should be visible within this window.

4. Once you have selected the desired image, click on the “Open” button located on the bottom left of the “Open JPEG File” dialog box (shown in figure 3). This will cause the image to be loaded into the Manipulator’s interface and both the picture and its data should be visible.

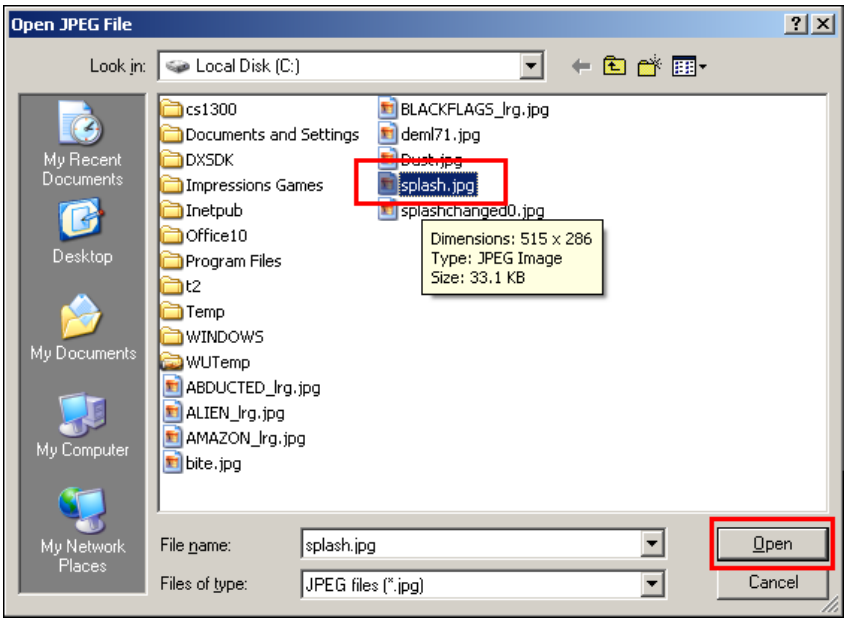


Figure 3: The "Open JPEG File" Dialog Box

Note: The image loaded here "splash.jpg" file included in the ISE folder (even though this picture shows the image in a different folder). Use this image to follow along with this tutorial.

5. Manipulating JPEG Image Data:

The most common function of the Manipulator (and the purpose for which it was designed) is to allow a user to manipulate data within a JPEG image and create a new image based upon the new data. Since there are many different pieces of a JPEG image, there are a large number of ways to make changes. Keep in mind that changing an image won't necessarily cause it to look different, only that the new image will contain the specified data. The following are detailed, step-by-step instructions for creating these ISE project files:

1. Once a JPEG image has been loaded in the Manipulator, click on the "Huffman Table 1" sub-tab to view the compression table data. This will cause the "Huffman Table 1" tab to be displayed, with any data that is included in the image (shown in figure 4). This data is represented by each individual byte's hexadecimal value. For example, if the table data says "0f," this means that particular byte has a value of 15 in base ten.

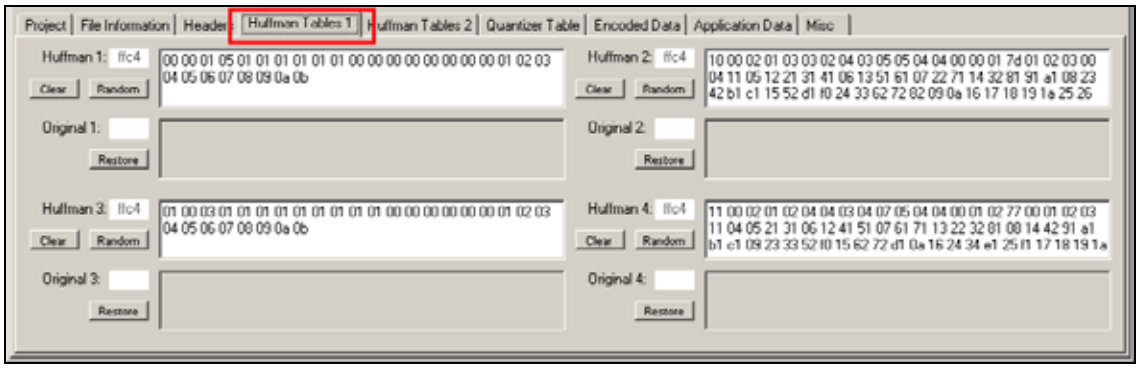


Figure 4: The "Huffman Tables 1" Sub-Tab

2. Then click on any of the text boxes where the individual tables data is displayed. When any of these text boxes are selected, the original data in the table will be stored just below in the grey text box below it. This is to ensure the original data in the image can always be restored in the future and can be done at any time by clicking on the corresponding "Restore" button. Change a few values within the table to begin the process of creating a manipulated image (shown in figure 5).

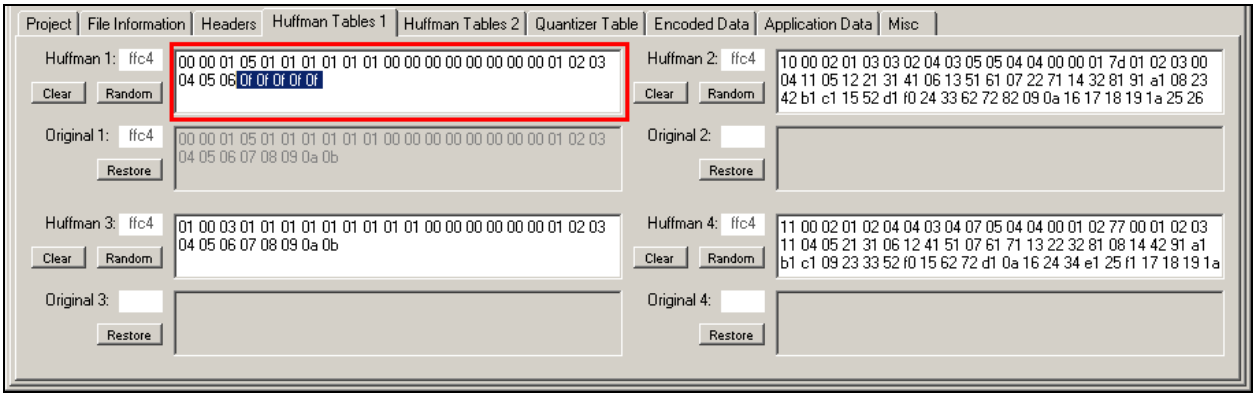


Figure 5: Manipulated Compression Table Data

3. Now that we've made changes to the original image data, we can actually create a new image. Click on the "File" menu option. This will cause the menu to become visible at the top of the window (shown in figure 6).

4. Then click on the "Update Picture" menu option (shown in figure 6). This will cause a new file JPEG file to be generated by the manipulator, based on any of the updated picture data. Once the new image has been generated, the Manipulator will attempt to load it into the "Manipulated Picture" boxes. Depending on the changes made to the image, it may or may not be viewable, but if not, then a default message image will be loaded instead. If the image is viewable, it may be the same or it may be distorted. The picture in figure 7 represents an example of how an image may look due to distortion caused by altering the image data.

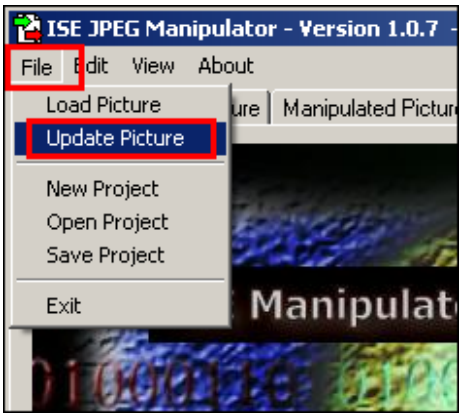


Figure 6: The "File" Menu (Update Picture Highlighted)

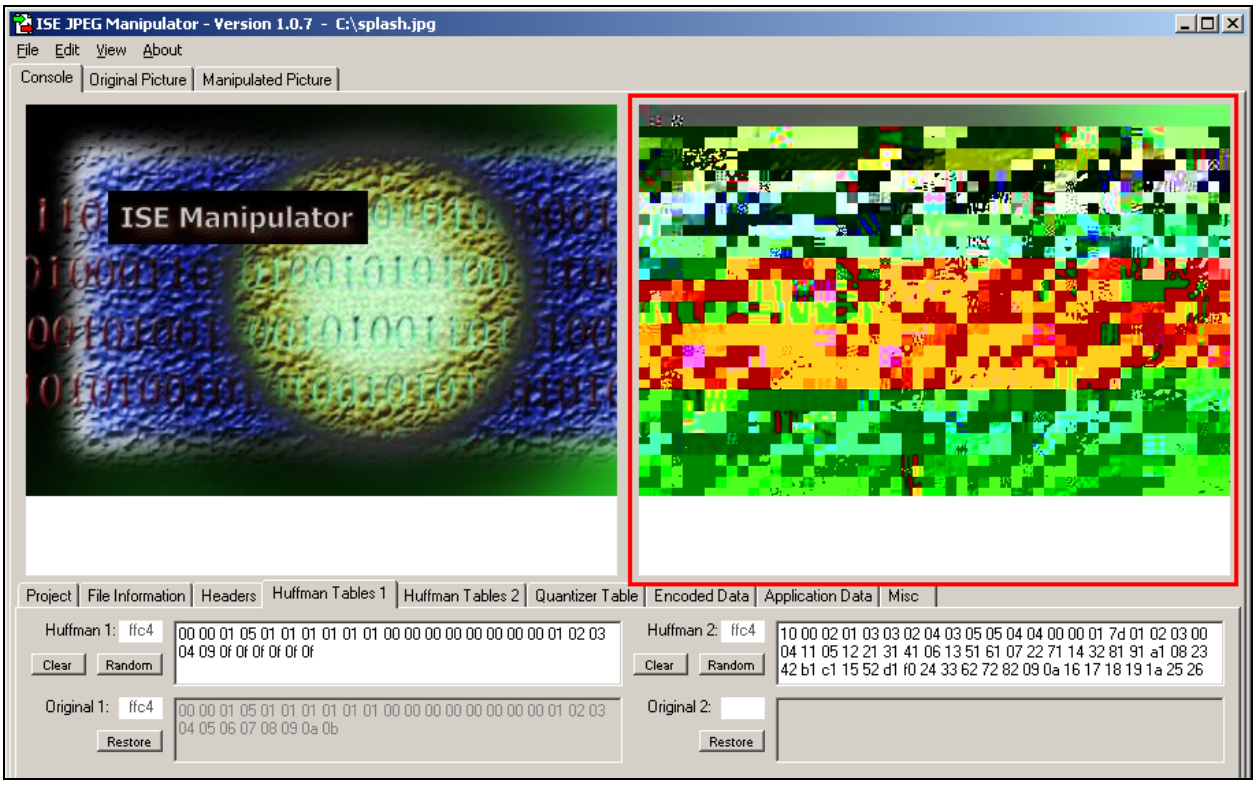


Figure 7: Example of an Manipulated Image

6. Creating an SEP Project File:

Another common function of the Manipulator is to allow the user to save information about a loaded image and any changes made to it in a project file. This project can then be loaded and updated at any time in the future, without losing data on an image that is currently a work in progress. There are two ways to do this within the application: on the main menu or on the "Project" sub-tab located on the "Console" tab. The following are detailed, step-by-step instructions for creating these ISE project files:

1. Click on the "File" menu option. This will cause the menu to become visible at the top of the window (shown in figure 8).

2. Then click on the "Save Project" menu option (shown in figure 8). This will cause the "Save SEP File" dialog box to appear (shown in figure 9).

3. Then create a new project by typing a name in the "File Name" text box or browse to an existing SEP file and click on it (shown in figure 9). Only images that have an SEP extension should be visible within this window.

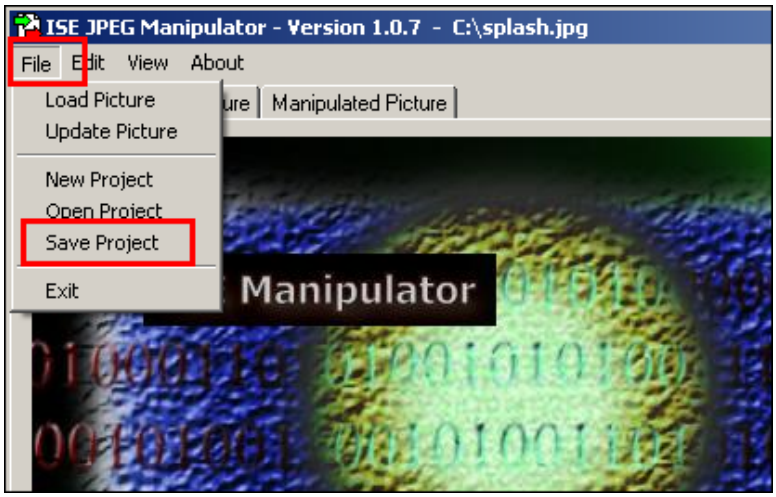


Figure 8: The "File" Menu (Save Project Highlighted)

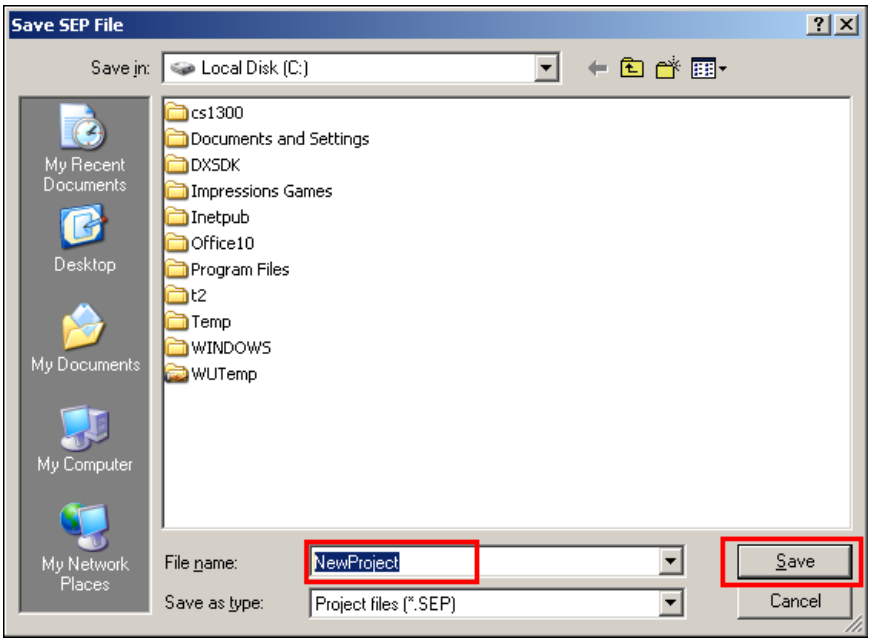


Figure 9: The "Save SEP File" Dialog Box

4. Once you have chosen the project name, click on the "Save" button located on the bottom left of the "Save SEP File" dialog box (shown in figure 9). This will cause all of the current data loaded in the Manipulator to be saved under the existing file name and path. Keep in mind that if you choose an existing file, all of the previous project file data will be overwritten with the new project information.

Closing Remarks:

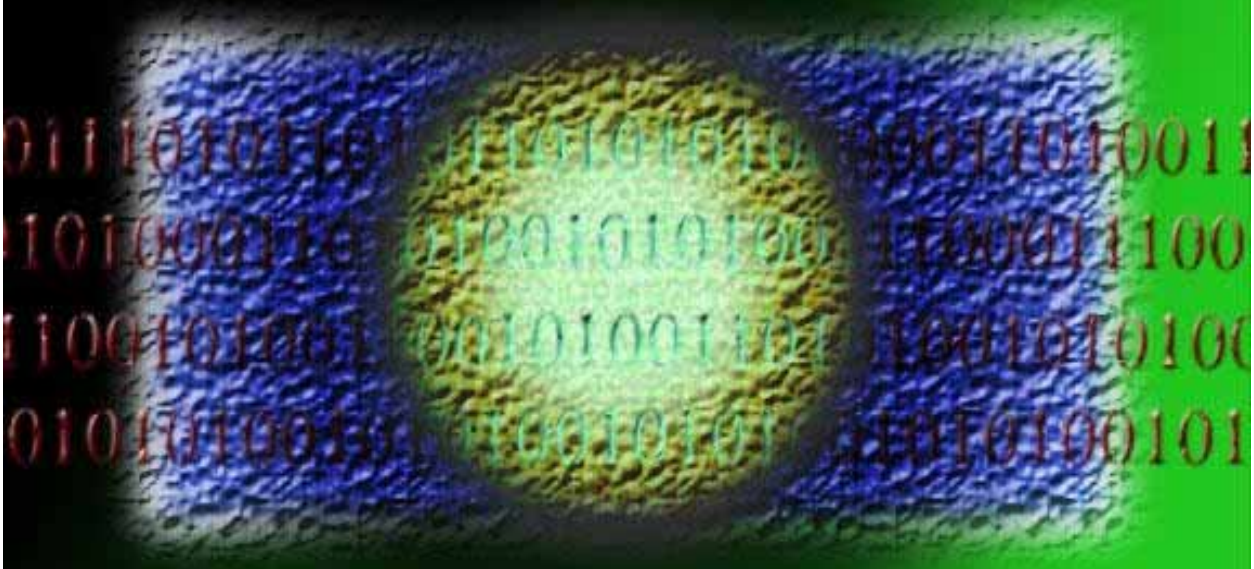
As you can see, the JPEG Manipulator is designed to be a straightforward, Windows-style application, to make the learning curve extremely easy. This tutorial was designed to give a basic overview of the Manipulator, but for more specific information about any of the functions more in-depth, please see the Manipulator User Manual. Also, for more information about the JPEG file format, please see the reference section of the ISE Manipulator Manual for documents pertain to this topic.

Manipulator
User Reference

Team ISE

Image Selective Encryption

Manipulator Reference Manual



Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Table of Contents

Introduction

The ISE JPEG Manipulator is an application designed to allow the user to examine, manipulate and create images from the data of pre-existing JPEG images. Team ISE, at the University of Colorado at Boulder, developed this software in conjunction with a research on the topic of JPEG Image Selective Encryption. The Manipulator was employed by the team to evaluate data from different JPEG images and provide support in testing different encryption schemes.

The purpose of this manual is to inform the user about the functionality of the JPEG Manipulator. Throughout the course of this manual, there will be a lot of terminology related to JPEG images. Unfortunately, this manual does not explain in-depth the about how JPEG compression formats work and assumes that the user has some prior knowledge of these concepts. If you do not have any previous experience with these types of compression formats, Team ISE recommends reviewing some of the reference material included at the end of this manual.

This software is provided by Team ISE and the University of Colorado AS IS and although it is believed to be in good working order, said members provide no guarantees and/or warranties about the Usage, Quality, Stability and/or Correctness of this software. Neither Team ISE nor the University of Colorado shall be held liable for any damages what-so-ever, arising out of or related to the use of or the inability to use the ISE JPEG Manipulator software. Anyone installing, using, or having previously installed or used the JPEG Manipulator agrees to all of these conditions. Any user should properly back-up all data before using it in conjunction with the ISE JPEG Manipulator software.

Chapter 1: Getting Started

Before the user can begin using the JPEG Manipulator, there are a number of tasks that must be accomplished. This section of the user manual is to explain the system requirements, provide installation instructions and opening the application. Please read over the following information for a description of these details.

1.1 ISE Manipulator Minimum System Requirements:

The Manipulator has several system requirements needed as a minimum to properly install and run the Manipulator. These requirements are as follows:

1. Microsoft Windows NT/2000/XP Operating System.
2. Microsoft .NET framework version 1.1.
3. A monitor, mouse and keyboard for the host computer.
4. 100 MB of free Hard Disk space.
5. 300 MHz or faster CPU.
6. 64 MB of RAM.

1.2 ISE Manipulator Recommended System Requirements:

In addition to the minimum system requirements needed for the ISE Manipulator, Team ISE also recommends several minimum system requirements, to ensure performance. These requirements are as follows:

1. 1 GHz or faster CPU.
2. 256 MB of RAM.

1.3 Installing the ISE JPEG Manipulator:

The Manipulator comes prepackaged with a full installation script that performs all necessary functions to properly install the Manipulator with a minimal degree of user effort. If a previous version of the Manipulator was installed prior to this installation, be sure to completely remove the previous version before preceding with this installation, otherwise this installation will not complete properly. Uninstalling the Manipulator is covered under section 1.4 of this manual.

If your system meets all minimum system requirements and there is no previous version of the Manipulator installed, you are now ready to begin the installation process. To install the ISE JPEG Manipulator, complete the following steps:

1. If the ISE Manipulator has not been previously downloaded, be sure to download it to the computer where it is to be installed to. Be sure to save it some place that can be easily remembered, like your “Desktop.”

2. Once the file has completed downloading, double-click on the file to begin the installation process.

3. Follow through on-screen installation instructions of the installer to successfully complete the installation (see figure 1.3.1). Team ISE recommends using all of the default settings for installation.

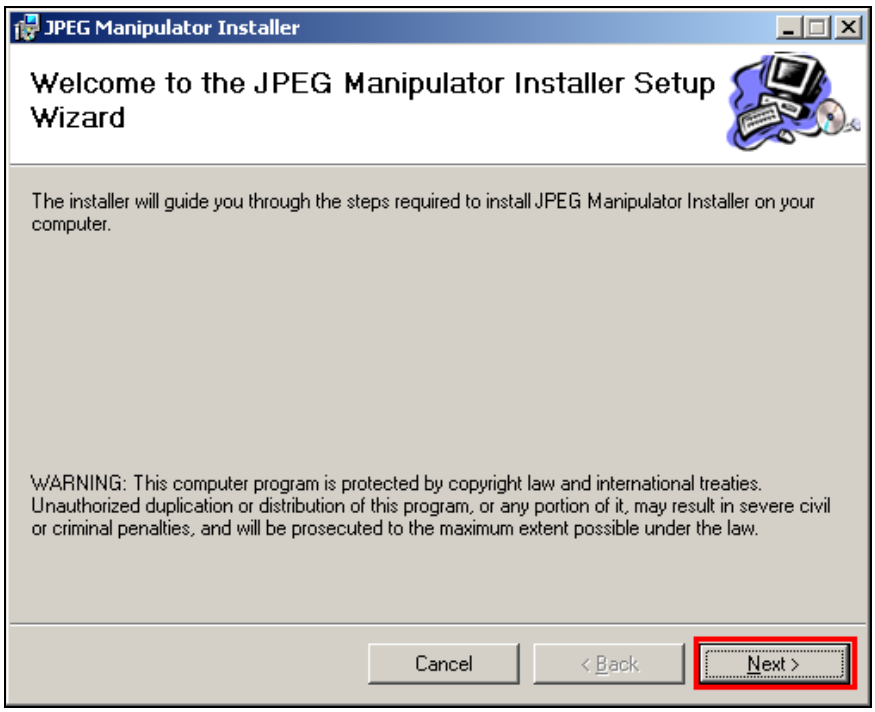


Figure 1.3.1 – JPEG Manipulator Installer Start Screen

1.4 Uninstalling the ISE JPEG Manipulator:

The Manipulator installation package also includes an uninstall script to remove all of the application files, if the need arises. The uninstaller will remove all data copied and created during the installation process. Please note that images created by the user or any original JPEG pictures used will not be removed, unless they are saved within the ISE program folder (the folder created in the programs files during installation). To perform the uninstall process, complete the following steps:

1. Go to Start >> Settings >> Control Panel

2. Once you click on the "Control Panel," you should see the contents of the Control Panel folder. Double-click on the "Add/Remove Programs" icon (See figure 1.4.1).

3. Within the Add/Remove Programs utility, find the "JPEG Manipulator" entry and click on it to highlight it in blue and then click on the "Remove" button next to it.

4. Follow through the on-screen instructions to successfully complete removal of the application.

Please note that you must uninstall any previous versions of the ISE JPEG Manipulator before installing any updated versions.

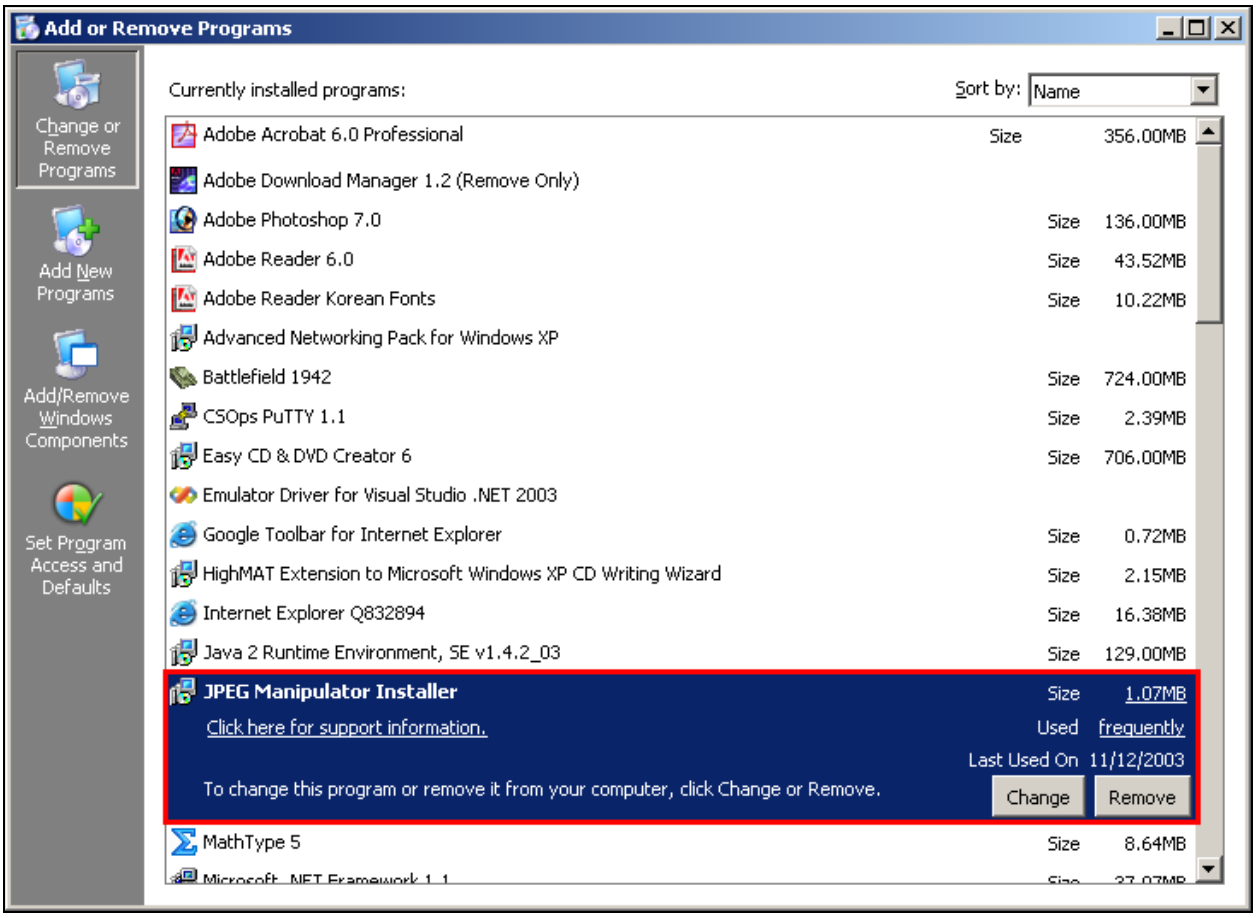


Figure 1.4.1 – Add/Remove Program Example

1.5 Running the ISE JPEG Manipulator Application:

Once the Manipulator has successfully completed installation, you should be able to instantiate the application without further delay. If the default installation was chosen, then the program can be invoked by going to:

Start Button >> Programs >> ISE >> JPEG >> JPEG Manipulator

Otherwise, if a different location for the program menu has been chosen, then go to the folder where the program was installed and then go to:

ISE >> JPEG >> JPEG Manipulator

Once the JPEG Manipulator icon has been clicked, the program will begin execution.

Chapter 2: JPEG Manipulator Functionality

Once you've successfully installed the ISE JPEG Manipulator application, you can begin to work with different types of JPEG images. Although the application is designed to work specifically with the Baseline compression standard for JPEG images, the application should work with other JPEG formats as well. This section of the manual is devoted to describing the interface and functionality of the JPEG Manipulator.

2.1 Understanding How Data is Represented in the Application:

Understanding the way JPEG image data is represented in the application is extremely important to use this as an effective tool. When the Manipulator loads a JPEG image, the data is broken down into each of the frames of data and then distributes to its corresponding text box on the interface. When each text box is loaded, the byte data is converted to its hexadecimal value in ASCII characters. For example, if a byte has the value of: 1111 1111 binary (0xff in Hexadecimal), then in the interface, the data will be displayed as "ff." If we look at an actual example of a Huffman table (shown in figure 2.1.1), we see that the data is loaded as 2 characters followed by a space for each byte of data in the table.

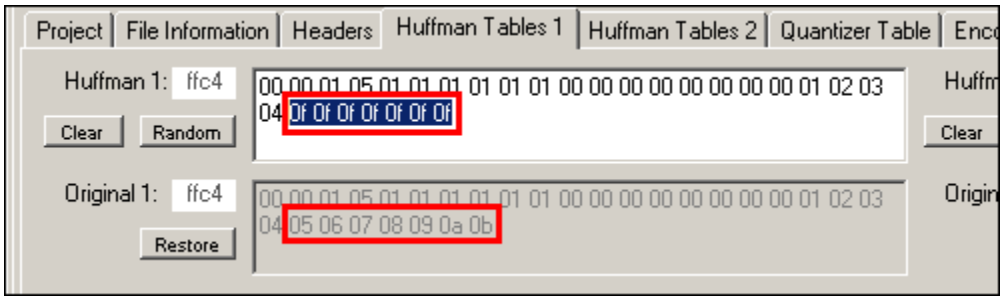


Figure 2.1.1 – Example Data in the Manipulator

As you can see by the picture above, there has been a change to 7 bytes of data in this particular Huffman table. Note that when we make a change to the data, we must represent it as a character of '0' to '9' or 'a' to 'f' for the Manipulator to properly interpret the data we have changed. Now that it has been said that data is represented in this way, we should mention that this is true for MOST of the text boxes, but not all. Certain text boxes, like the File Comments text box, outputs the data as its byte value, which allows the user to see what the data actually says, not just the byte data itself. In the following sections, each of the different text fields will be defined as to which way data is represented. In most cases, the text fields will contain the byte data converted to the ASCII representation of its value.

2.2 The Manipulator's Main Window:

Upon invoking the JPEG Manipulator application, the main window will be displayed for the user to see. The main window is also known as the Console tab (which is described in depth in section 2.6) and will provide the user access to almost every function of the Manipulator. The main window is designed to look and feel like a standard, Windows-style application. Notice that there is a main menu, tab controls to switch between the different components of the interface and closing and sizing controls for the window itself.

2.3 The Main Menu of the Manipulator:

The JPEG manipulator contains a Windows-style application menu bar for performing common tasks within the Manipulator. This main menu consists of four sub-menus: File, Edit, View and Help sub-menus. Each of these sub-menus is defined in this section of the manual.

2.3.1 The File Menu:

The File menu is the leftmost menu on the main window of the JPEG Manipulator application. This menu provides the user with options to Load a Picture, Update a Picture, Create a New Project, Save a Project, Load a Project and Exit the application. Please note that all of these functions (except exiting the application) can be preformed on the Project sub-tab by using the buttons available. In addition, the application can also be exited at any time by clicking on the "X" button on the top right of the main window of the application. Figure 2.3.1 shows an example of the File menu and the following is a description of each of the File menu options:

1. **Load Picture** - Allows the user to Load a new JPEG image into the manipulator as an original image for editing. If there is an existing picture previously open, the user will be warned before the Load Picture action is taken.
2. **Update Picture** - Allows the user to create a new manipulated image, based upon the current data in the Manipulator data fields.
3. **New Project** - Clears out the current project data and picture data in the Manipulator.
4. **Open Project** - Allows the user to open an existing project. If a project is already open, then the user will be warned before the Open Project action is taken.
5. **Save Project** - Allows the user to save all of the data currently loaded in the Manipulator in an SEP project file for future use.
6. **Exit** - Allows the User to close the JPEG Manipulator at any time.

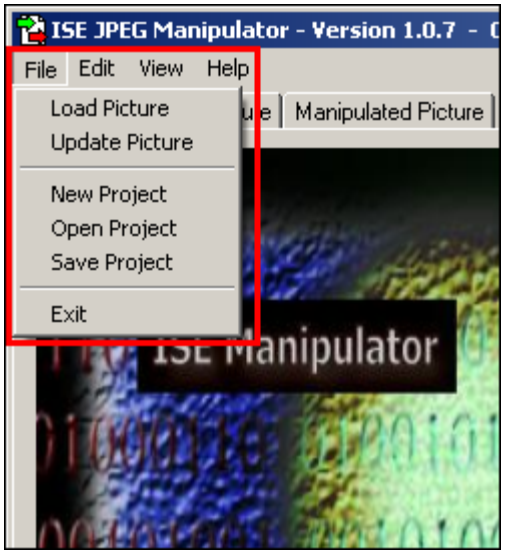


Figure 2.3.1 – The File Menu

2.3.2 The Edit Menu:

The Edit menu is the second from the leftmost menu on the main window of the JPEG Manipulator application. This menu provides the user with options to Copy, Cut and Paste data to and from the system clipboard. Please note that all of these functions can be preformed by keying in the combinations: 'ctrl+c', 'ctrl+x' or 'ctrl+v' respectively. Figure 2.3.2 shows an example of the Edit menu and the following is a description of each of the Edit menu options:

1. **Copy** - Copies the currently highlighted text to the system clipboard.
2. **Cut** - Cut the currently highlighted text and copies it to the system clipboard.
3. **Paste** - Pastes any text currently on the system clipboard into the field currently indicated by the text cursor.

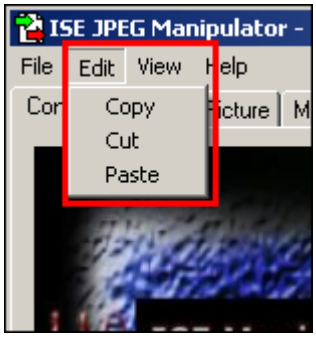


Figure 2.3.2 – The Edit Menu

2.3.3 The View Menu:

The View menu is the second from the rightmost menu on the main window of the JPEG Manipulator application. Under the View menu is the Stretch Mode sub-menu. This sub-menu provides the user with options to toggle each of the picture box controls between normal and stretch mode. Figure 2.3.3 shows an example of the View menu and Stretch Mode sub-menu and the following is a description of each of the options:

1. **Large Original** - Toggles the stretch mode for the picture box on the Original Picture tab.
2. **Large Manipulated** - Toggles the stretch mode for the picture box on the Manipulated Picture tab.
3. **Small Original** - Toggles the stretch mode for the Original picture box on the Console tab.
4. **Small Manipulated** - Toggles the stretch mode for the Manipulated picture box on the Console tab.
5. **All Pictures** - Toggles the stretch mode for each of the picture box controls in the Manipulator application.

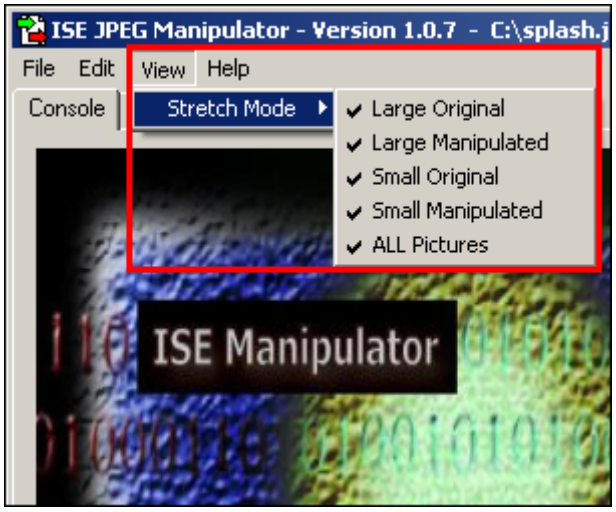


Figure 2.3.3 – The View Menu

2.3.4 The Help Menu:

The Help menu is the rightmost menu on the main window of the JPEG Manipulator application. This menu provides the user with options to view the Manipulator Tutorial, Manipulator User Manual (this document) and the About window. Figure 2.3.4 shows an example of the Help menu and the following is a description of each of the Help menu options:

1. **Tutorial** - Opens a new window containing the Manipulator's user tutorial document.
2. **Manual** - Opens a new window containing the Manipulator's user manual document (i.e. this document).
3. **About** - Opens a new window containing the Manipulator's about window to display the application and project information.

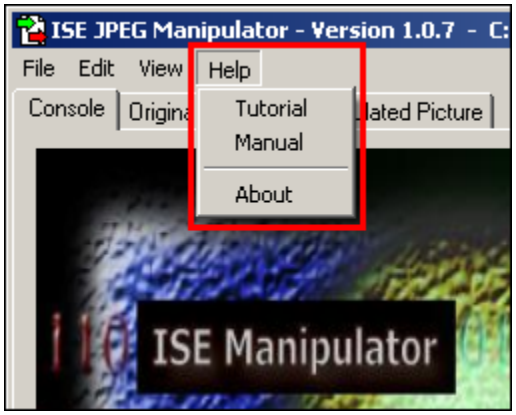


Figure 2.3.4 – The Help Menu

2.4 The Original Picture Tab:

The Original Picture tab allows the user to see, in a large window, the JPEG image that is currently loaded in the Manipulator. Also, this picture can be viewed in its actual size, or stretched to fit the exact size of the picture box control on this tab, by making changes in the

View menu settings (as described in section 2.3). An example image is shown in figure 2.4.1 (shown in stretch mode):

Figure 2.4.1 – The Original Picture Tab

2.5 The Manipulated Picture Tab:

The Manipulated Picture tab allows the user to see, in a large window, the altered JPEG image that is currently loaded in the Manipulator. Also, this picture can be viewed in its actual size, or stretched to fit the exact size of the picture box control on this tab, by making changes in the View menu settings (as described in section 2.3). An example image is shown in figure 2.4.1 (shown in stretch mode):

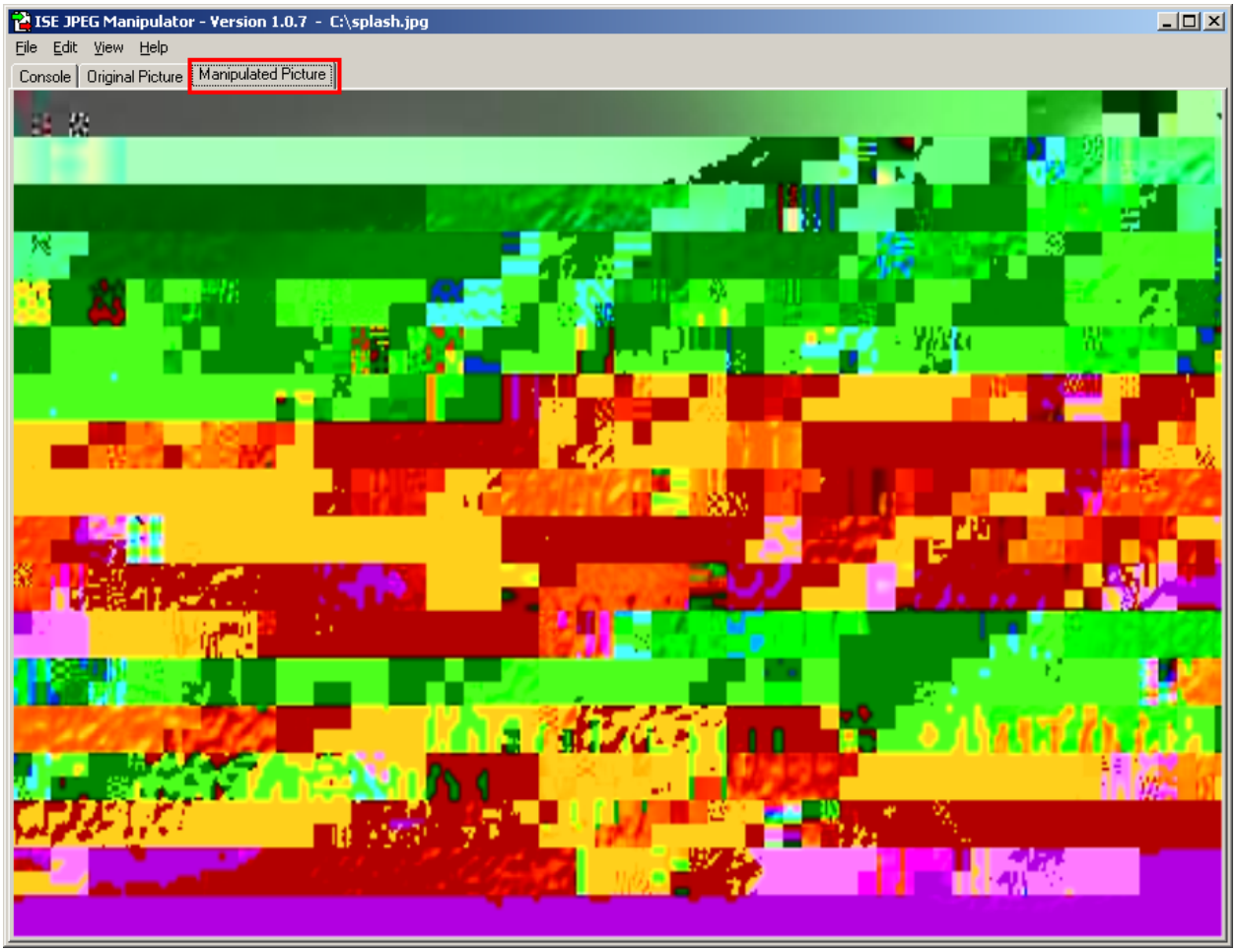


Figure 2.5.1 – The Manipulated Picture Tab

2.6 The Console Tab:

The Console tab is the real heart of the JPEG Manipulator. From the Console tab, the user can load images and projects, manipulate the data of images, create new images and save project data. The Console tab provides two smaller picture boxes for viewing the original and manipulated images, which can also be stretched by adjusting the settings on the View menu. The Console tab also has a series of sub-tabs that contain a number of text fields that store all of the image and project data. An example of the Console tab is shown in the figure 2.6.1 (note that the small original and manipulated pictures are in stretch view mode).

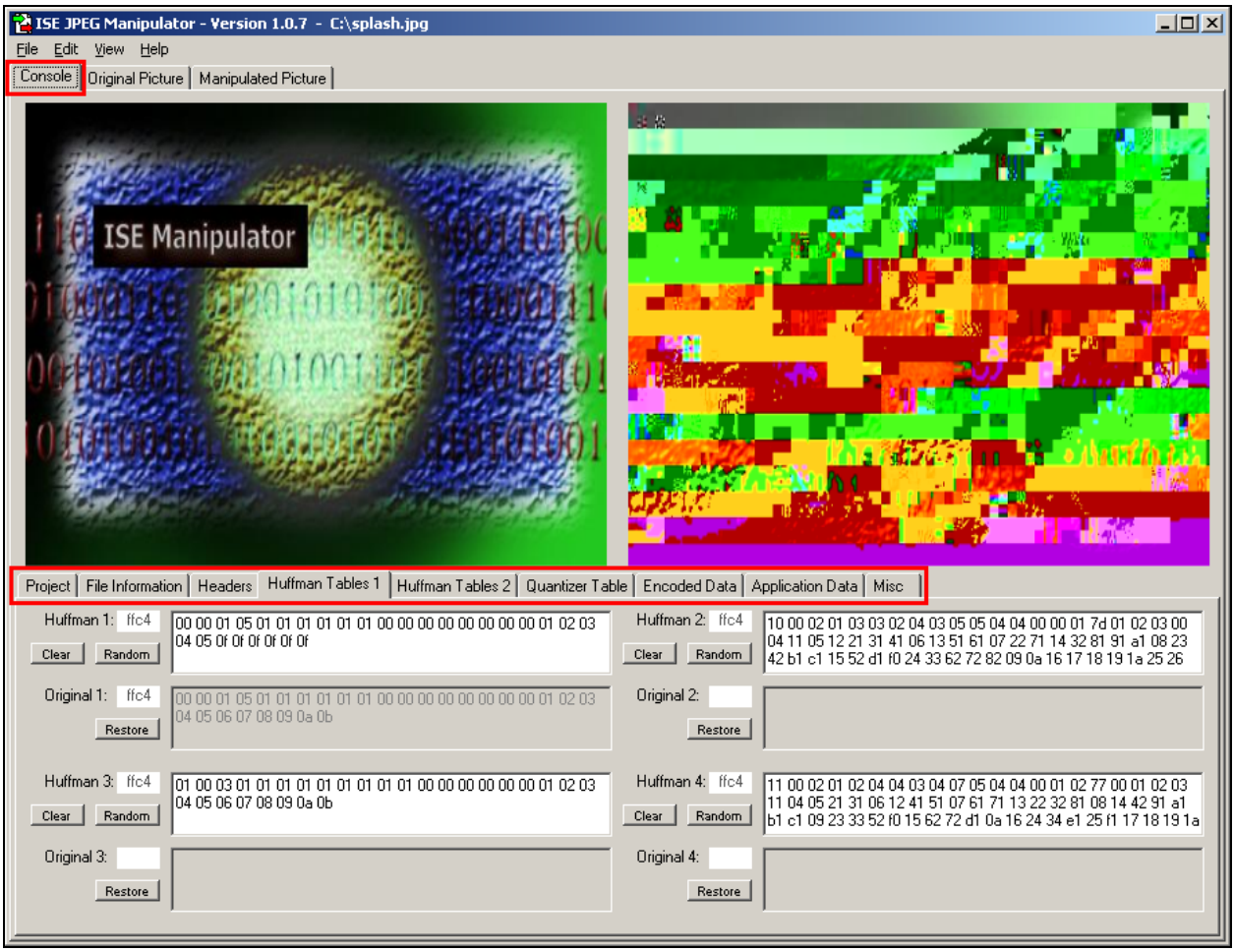


Figure 2.6.1 – The Console Tab

2.7 The Console Tab JPEG Data Sub-Tabs:

As shown in the picture above, the Console tab has a number of sub-tabs that contain all of the data loaded from the JPEG image and any project information. From these data sub-tabs, it is possible to change any of the data within the JPEG image. Each of these tabs are described in detail in this section of the manual. As stated in section 2.1, the data is represented a few different ways, depending on the type of image data that it happens to be. The following subsections will specify how the data for each of the data fields in the Manipulator is handled the data and how it will be treated when a manipulated image is encoded.

2.7.1 The Project Sub-Tab:

The Project sub-tab allows the user to view the path and file name of the current project loaded (if any) and any project notes. Notice that the Project file name and the Project notes each have their own text fields. In addition, you can see six buttons here, which perform some of the functionality that can be found on the main menu (as described in section 2.3). Figure 2.7.1 is an example of what Project sub-tab looks like:

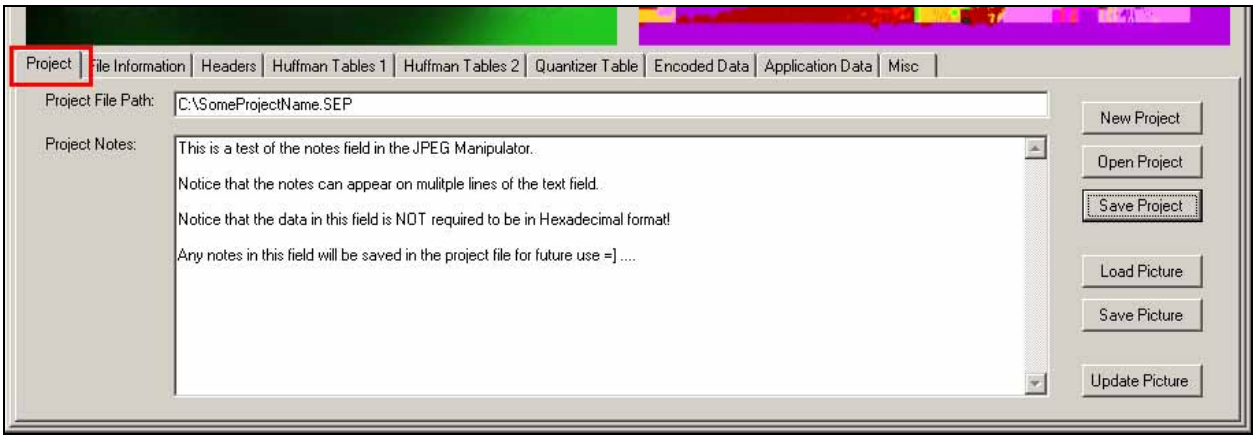


Figure 2.7.1 – The Project Sub-Tab

Notice that data stored in both of these fields can have any ASCII value and are not required to be entered as the hexadecimal characters as explained in section 3.1 of this document. Each of the six buttons found on this sub-tab performs the each of the respective functions:

1. **New Project** - Clears out the current project data and picture data in the Manipulator.
2. **Open Project** - Allows the user to open an existing project. If a project is already open, then the user will be warned before the Open Project action is taken.
3. **Save Project** - Allows the user to save all of the data currently loaded in the Manipulator in an SEP project file for future use.
4. **Load Picture** - Allows the user to Load a new JPEG image into the manipulator as an original image for editing. If there is an existing image previously open, the user will be warned before the Load Picture action is taken.
5. **Save Picture** - Allows the user to save the currently manipulated picture for future use.
6. **Update Picture** - Allows the user to create a new manipulated picture, based upon the current data in the manipulator data fields.

2.7.2 The File Information Sub-Tab:

The File Information sub-tab contains the file names of both the original image and the manipulated image, the size of the original file and the file comments in the original file. The data contained in these fields can be any ASCII character and is not limited to Hexadecimal format, as outlined in section 3.1 of this document. Also, any changed made to the data in the file comments will not be applied when the Manipulated image is generated. Figure 2.7.2 is an example of the File Information sub-tab:

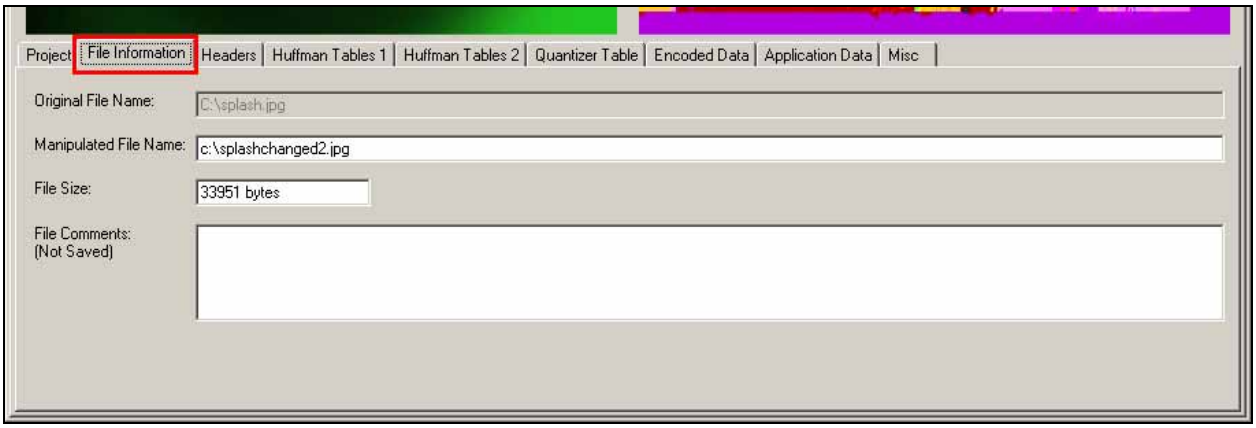


Figure 2.7.2 – The File Information Sub-Tab

2.7.3 The Header Data Sub-Tab:

If the current image loaded is indeed a Baseline standard compression format, then the compression table header data will be broken up and loaded into the fields on the Header sub-tab. The information displayed here will be the SOF0 frame data (which should always be the ffc0 marker), the size of the data contained in the frame, the precision data, height and width data, the number of components and finally the component definitions. If the image loaded is not the Baseline standard compression format, then the Headers tab shouldn't contain any of the JPEG image data.

In addition, it is important to understand that the data on the Header sub-tab is represented as its Hexadecimal value. This type of representation is outlined in section 3.1 of this document and any changes made to these fields should be made in the same way, as the hexadecimal format of the values you wish to change the data to. For example, the Marker field here should say "ff c0" which means that the binary value of these two bytes in the file is: "1111 1111 0110 0000." Also, any changes in the size of the data in this frame does not need to be made by the user manually as the Manipulator will calculate the new size of the frame (and number of components) when generating the manipulated image. The picture below is an example of the Header data sub-tab:

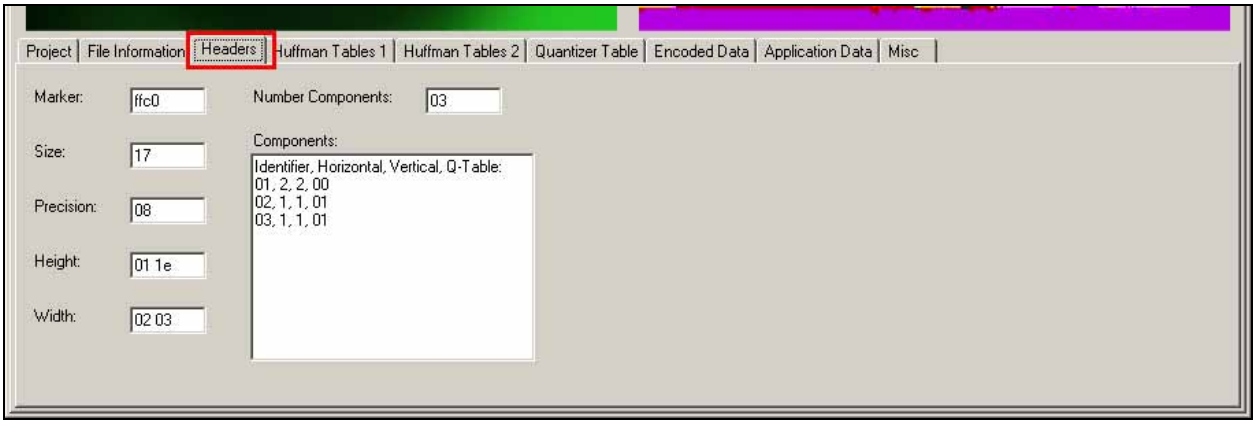


Figure 2.7.3 – The Headers Sub-Tab

2.7.4 The Huffman Table Sub-Tabs:

As mentioned previously, the Manipulator was designed specifically for use with the Baseline standard JPEG compression format. This format uses the Huffman compression algorithm to create the entropy encoded data stream for the JPEG image. Thus, these compression tables can be found on the two Huffman Table sub-tabs. However, even if the image doesn't use the baseline compression, any compression data encountered in the JPEG file will be stored on this sub-tab.

The Huffman Table sub-tabs contain a number of text fields and buttons for the user to interact with the image data. Each tab contains data fields for four tables, original and manipulated compression table data, the compression table maker data, and a series of buttons for manipulating the tables. When a compression frame is loaded in the manipulator, the marker is loaded in the marker field, and then the rest of the frame's data, except for the size information, is loaded into the corresponding Huffman table field. The data is loaded in Hexadecimal format and any changes made to this data should be as hexadecimal ASCII characters that represent the values of each of the bytes of the data. Section 3.1 describes in-depth how this data is represented. Also, the size of the marker is purposefully removed from this table so that if the user changes the size of the table, then the new size data will be recalculated when the new image is generated. Notice that once the user tries to manipulate the values of one of the tables, the original data is stored in the corresponding, grayed out field just below it. There are three buttons corresponding to each of these tables:

1. **Clear** - this button clears out all of the data in the corresponding table field.
2. **Random** - this button adds a random byte of data onto the end of the field.
3. **Restore** - this button restores the original data from the image to the corresponding compression table.

Figure 2.7.4 is an example of the Huffman Table 1 data sub-tab:

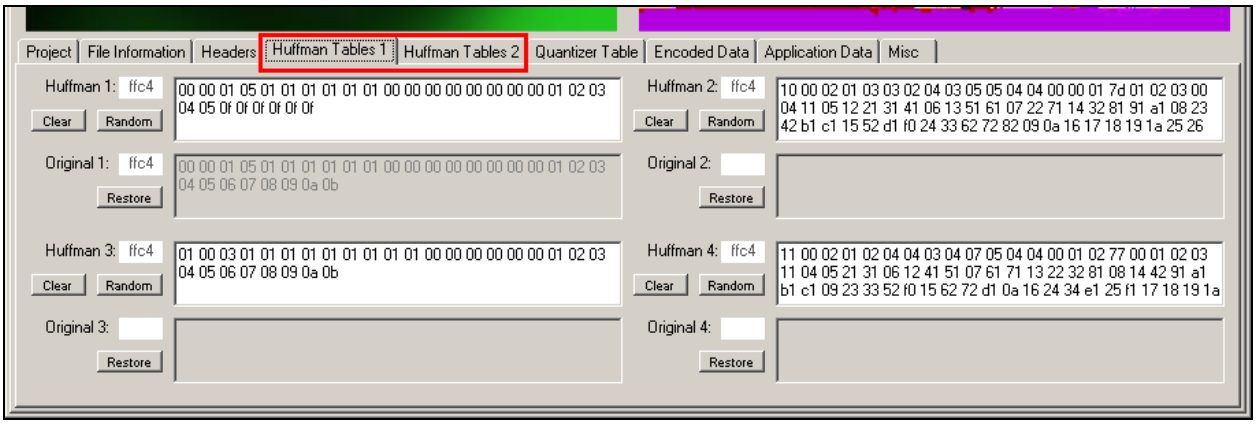


Figure 2.7.4 – The Huffman Tables Sub-Tabs

2.7.5 The Quantizer Table Sub-Tab:

The Quantizer table sub-tab contains all of the data for the DQT frames (i.e. the Quantizer tables) contained in the JPEG image. As defined by the JPEG standard, each of these tables should be exactly 64 bytes long for all Baseline compression images. Each of the Quantizer's marker data, table data and table number is displayed here. In addition, there are several buttons to interact with the data.

When the Quantizer frame is loaded in the manipulator, the marker is loaded in the marker field, and then the rest of the frame's data, except for the size information and the table information (i.e. all of the DTC coefficients) are loaded into the corresponding Quantizer table field. The data is loaded in Hexadecimal format and any changes made to this data should be as hexadecimal ASCII characters that represent the values of each of the bytes of the data. Section 3.1 describes in-depth how this data is represented. Also, the size of the frame is purposefully removed from this table so that if the user changes the size of the table (even though the Baseline standard dictates that these tables have exactly 64 coefficients), then the new size data will be recalculated when the new image is generated. Notice that once the user tries to manipulate the values of one of the tables, the original data is stored in the corresponding, grayed out field just below it. The three buttons corresponding to each of these tables has the following function:

1. **Clear** - this button clears out all of the data in the corresponding table field.
2. **Random** - this button adds a random byte of data onto the end of the field.
3. **Restore** - this button restores the original data from the image to the corresponding Quantizer table.

The picture below is an example of the Quantizer Table sub-tab:

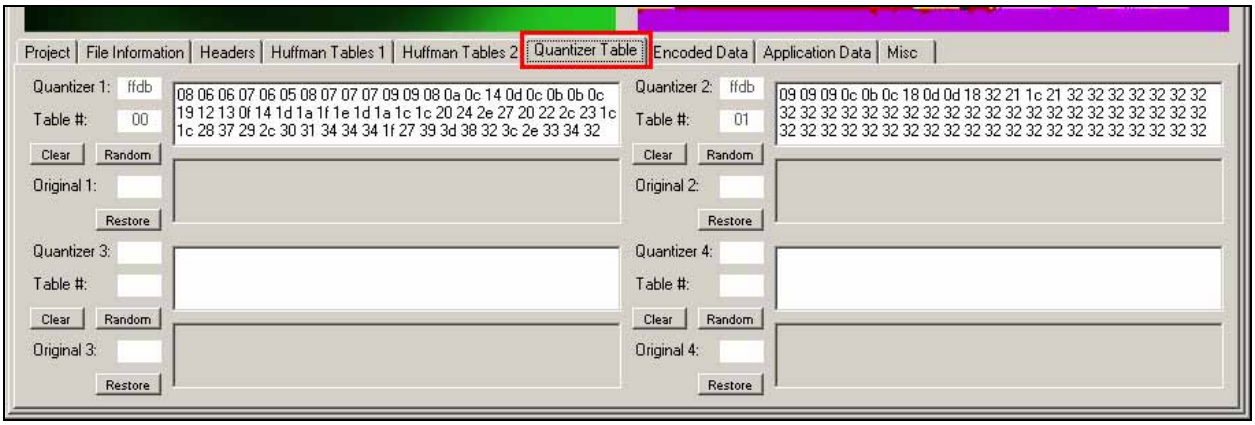


Figure 2.7.5 – The Quantizer Table Sub-Tab

2.7.6 The Encoded Data Sub-Tab:

The Encoded Data sub-tab contains all of the SOS frame data (i.e. Scan Header data), as well as up to the first 20 KB of the entropy encoded data stream. Although there are fields for the original data, any changes made to the data on this sub-tab will not be applied to the Manipulated picture as this functionality will have to be left for a future

software enhancements. Therefore, any changes made on the text fields will not be accounted for.

When the Scan Header frame and the Entropy Encoded data is loaded in the manipulator, everything except the frame marker is stored on Encoded Data sub-tab. The data is loaded in Hexadecimal format, although any changes made to this data will not be accounted for. Section 3.1 describes in-depth how this data is represented. Also, keep in mind that only the first 20 KB of the Encoded Data is shown in the encoded data field. Figure 2.7.6 is an example of Encoded Data sub-tab:

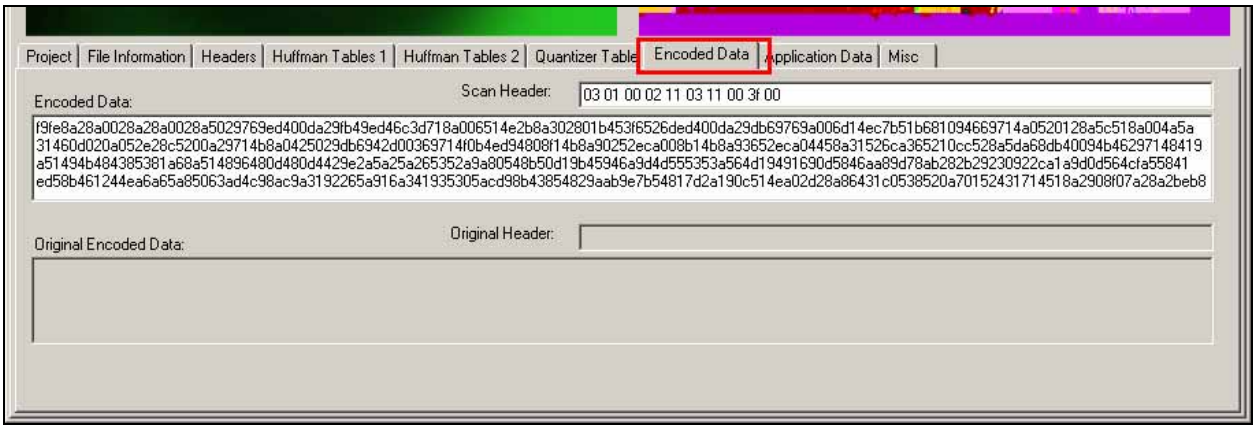


Figure 2.7.6 – The Encoded Data Sub-Tab

2.7.7 The Application Data Sub-Tab:

The Application Data sub-tab contains all of the data found in APP frames (i.e. Application Data frames) of the JPEG image. Although there are up to sixteen application frames allowed in the Baseline compression standard, there are only ten spaces available on the Application data tab. Since application data isn't critical to the representation of the particular image, the decision was made to only include up to ten frames in the Manipulator (in all the random samples collected during the development of the Manipulator, none contained more than ten, in fact all of the images contained much less).

When the Application Data frame is loaded in the manipulator, the marker is loaded in the Marker field, and then the rest of the frame's data, except for the size information is loaded into the corresponding text field. The data is loaded in Hexadecimal format and any changes made to this data should be as hexadecimal ASCII characters that represent each of the bytes of the data. Section 3.1 describes in-depth how this data is represented. The size of the frame is purposefully removed from this table so that if the user changes the size of the table, then the new size data will be recalculated automatically when the new image is generated. Figure 2.7.7 is an example of the Application Data sub-tab:

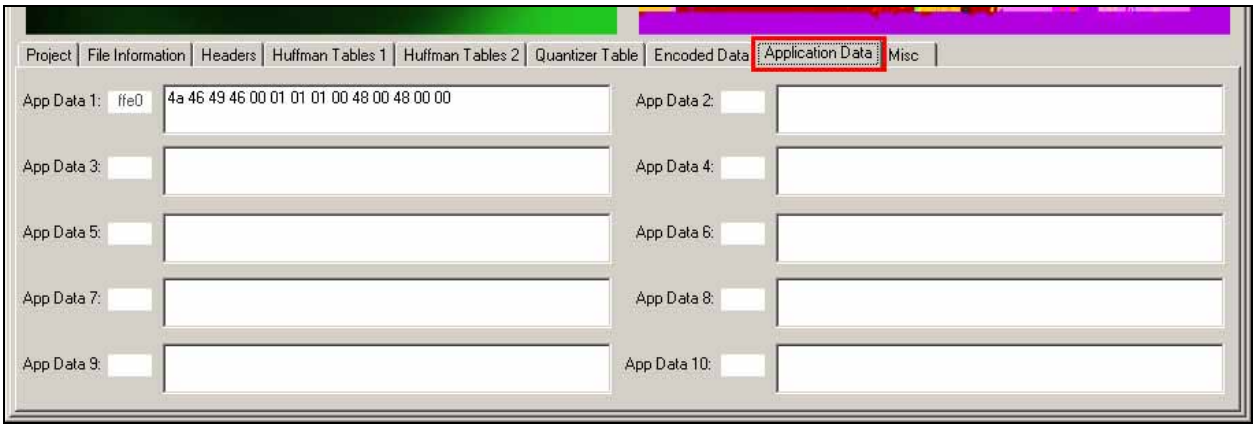


Figure 2.7.7 – The Application Data Sub-Tab

2.7.8 The Misc Data Sub-Tab:

The Misc sub-tab contains the data of the rest of the possible frames allowed within a JPEG image. Much of this data is not allowed for the Baseline compression standard, but to make sure that the Manipulator would be compliant for all standards currently defined, these data fields were included. The Misc sub-tab contain data fields for the marker data and frame data for the restart interval frame, marker data and frame data for the number of lines frame, marker data and frame data for the expand image frame, marker data and frame data for any hierarchical progression frame, data for the Restart Mod 8 frame and a field for any program errors generated by the Manipulator during execution.

When the frames on the Misc sub-tab are loaded in the Manipulator, the marker is loaded in the marker field (except for the Restart Mod 8 field), and then the rest of the frame's data, except for the size information is loaded into the corresponding text field. The data is loaded in Hexadecimal format and any changes made to this data should be as hexadecimal ASCII characters that represent the values of each of the bytes of the data. Section 3.1 describes in-depth how this data is represented. The size of the frame is purposefully removed from these tables so that if the user changes the size of the table, then the new size data will be recalculated automatically when the new image is generated. The picture below is an example of the Misc Data sub-tab:

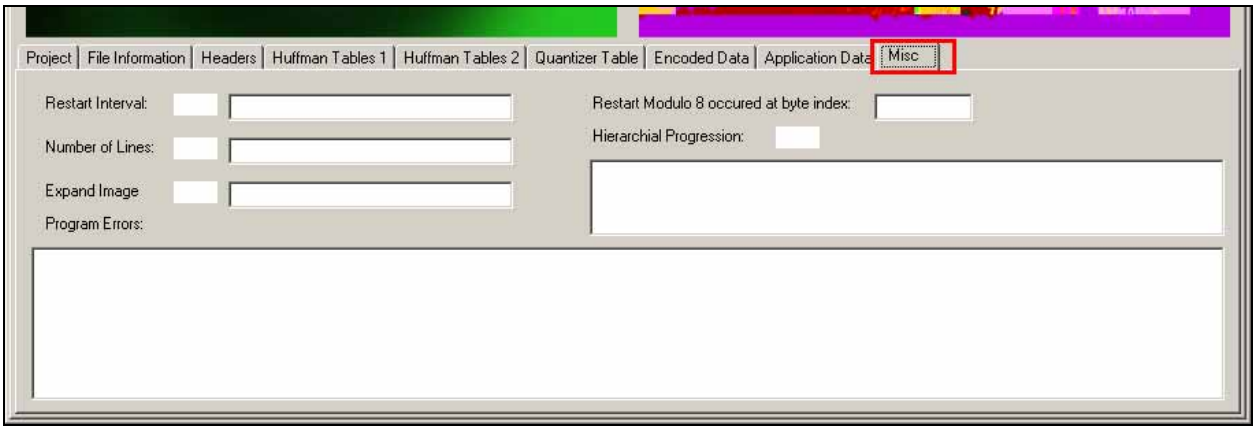


Figure 2.7.8 – The Misc Sub-Tab

Chapter 3: References and Related Readings

There were a number of documents and research papers used by Team ISE in completing this piece of software and this entire project. This section of the document is devoted to giving credit to all of those extremely helpful references. Team ISE recommends looking over the following data if you are unfamiliar with the JPEG file formats or the concept of Selective Encryption.

Chang, H. and Li, X. On the Application of Image Decomposition to Image Compression and Encryption. Research Paper. 1996.

Chang, H. and Li, X. Partial Encryption of Compressed Images and Videos. Research Paper. 2000.

Droogenbroek, M. and Benedett, R. Techniques for Selective Encryption of Uncompressed and Compressed Images. Research Paper. 2002.

Kailasanathan, C. and Naini, R. Compression Performance of JPEG Encryption Scheme. Research Paper. 2003.

Daigaku, S., et al. Requirement Specification. 2003.

Daigaku, S., et al. Selective Encryption of JPEG Standard Baseline Compression Images. 2004.

Daigaku, S., et al. System Architecture. 2003.

Li, X., Knipe, J. and Cheng, H. Image Compression and Encryption Using Tree Structures. Research Paper. 1997.

Lookabaugh, T., et al. Security Analysis of Selectively Encrypted MPEG-e Streams. Research Paper. 2003.

Miano, J. Compressed Image File Formats. Massachusetts: Addison Wesley Longman, Inc., 1999.

Norcen, R. and Uhl, A. Selective Encryption of the JPEG2000 Bitstream. Research Paper. 2003.

Pennebaker, W. and Mitchell J. JPEG Still Image Data Compression Standard. New York: Van Nostrand Reinhold, 1993.

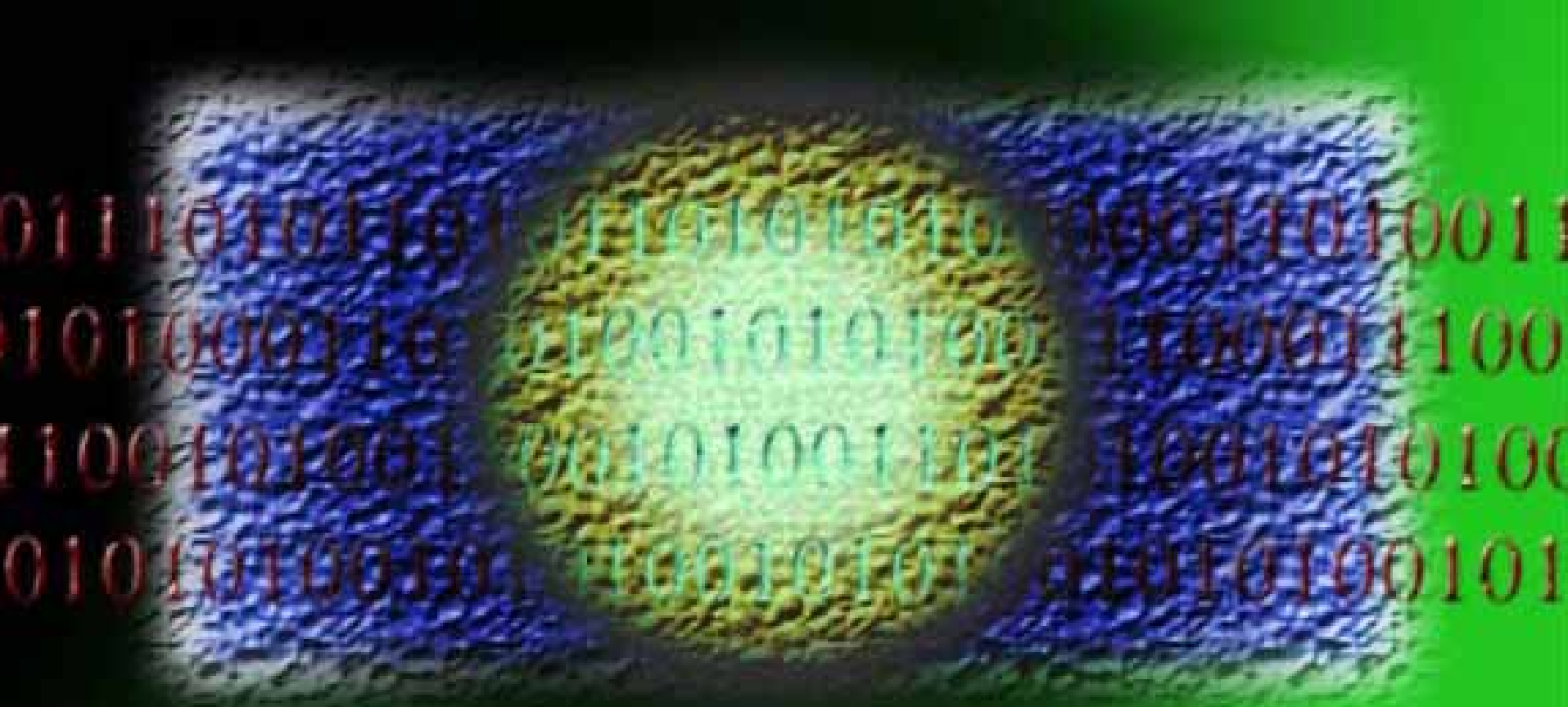
Podesser, M., Schmidt, H. and Uhl, A. Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments. Research Paper. 2002.

Seo, Y., et al. Wavelet Domain Imag Encryption by Subband Selection and Data Bit Selection. Research Paper. 2003.

Final Demo Presentation

Team ISE

Image Selective Encryption



Team ISE

Image Selective Encryption

Team ISE

Image Selective Encryption

Joe Jarchow

Joseph Kadhim

Geoffrey Griffith

Shinya Daigaku

Andrew Pouzeshi

Presentation Overview:

- **Overview of Project**
- **Demonstration of**
  - **Manipulator**
  - **Production Code**
  - **Web Site**
- **Algorithm Design**
- **Potential Attacks**
- **Conclusion**

Presentation Overview:

- Overview of Project
- Demonstration of
  - Manipulator
  - Production Code
  - Web Site
- Algorithm Design
- Potential Attacks
- Future efforts

Problem:

- **Multimedia files are very large**
- **Encryption is expensive**
  - **Processing time**
  - **File size**
- **No widely accepted solutions**
  - **Encrypt entire file**
  - **No encryption**

Solution:

•Selective Encryption

Definition from MPEG paper:

Selective encryption applies encryption to a subset of a file with the expectation that the entire file will be rendered useless to anyone who cannot decrypt that subset.

Selective Encryption Requirements:

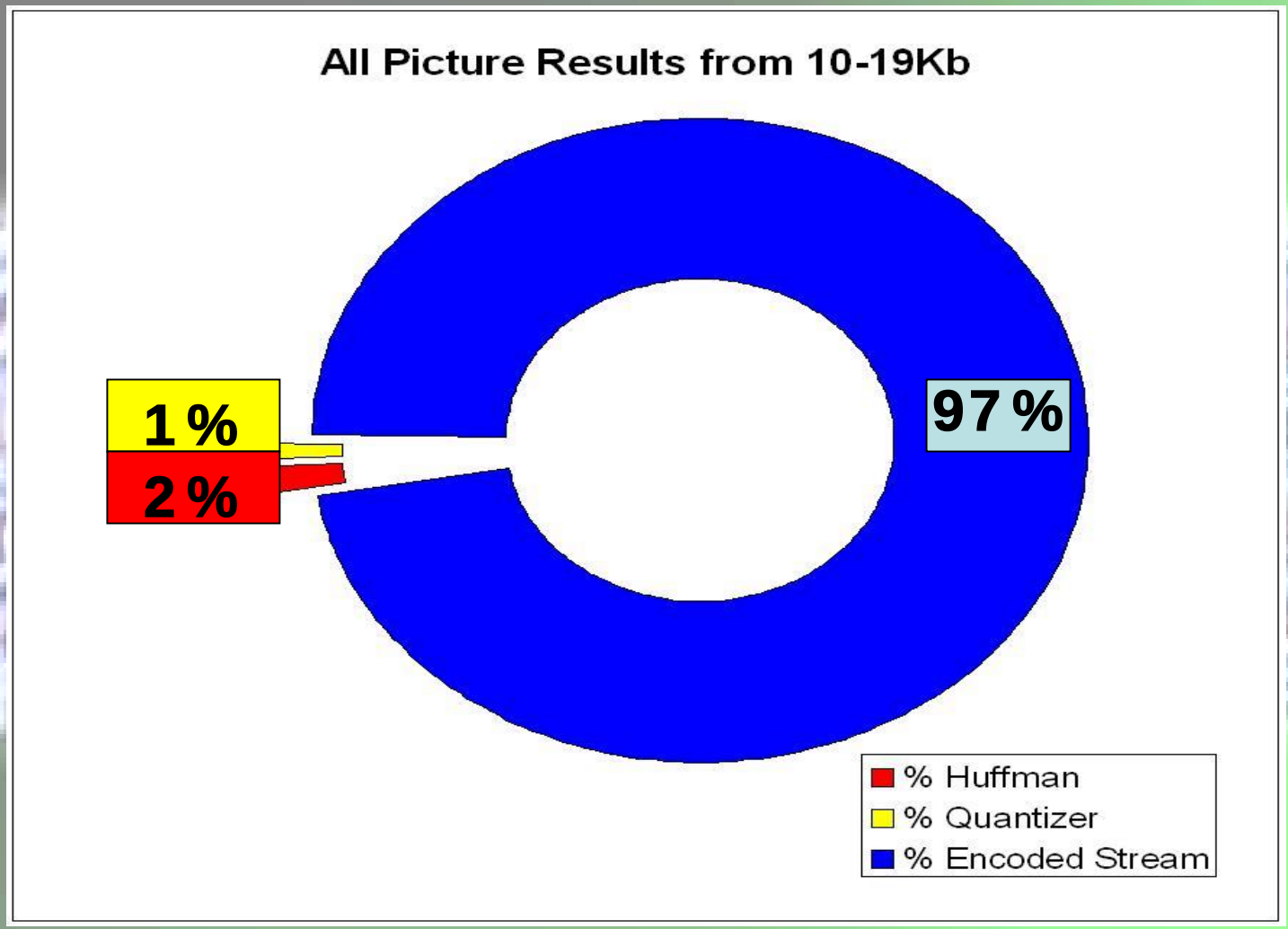
- **Perceivable degradation of file**
- **Encryption of less than 10 %**
- **Minimize required computation**
- **Minimize increase in file size**
- **Cryptanalytic approach**

JPEG Requirements

- **Only Baseline standard**

Criteria For Bad Targets:

- **Optional markers**
- **Not used in Baseline JPEG images**
- **No effect on image quality**
- **Easily guessed or forged by a hacker**


All Picture Results from 10-19Kb
1 %
2 %
97 %
% Huffman
% Quantizer
% Encoded Stream

Presentation Overview:

- Overview of Project
- Demonstration of
  - Manipulator
  - Production Code
  - Web Site
- Algorithm Design
- Potential Attacks
- Conclusion

Demonstration of Manipulator:

- **Layout vs. JPEG standard**
- **Show new features (project file, etc.)**
- **Cover earlier research**
- **Propose possible attacks**
- **Show table manipulation**
- **Show table replacement**

Presentation Overview:

- Overview of Project
- Demonstration of
  - Manipulator
  - Production Code
  - Web Site
- Algorithm Design
- Potential Attacks
- Conclusion

Demonstration of Production Code:

- **Run demonstration script**
- **During run, show code (h, cpp, scripts)**
- **Explain tests run in script (diff)**
- **Show images for comparison**
- **Show .ise will not work**

Presentation Overview:

- Overview of Project
- Demonstration of
  - Manipulator
  - Production Code
  - Web Site
- Algorithm Design
- Potential Attacks
- Conclusion

Demonstration of Manipulator:

- **Show menu bar links**
- **Show each page**
- **Show message board**
- **Show message board administration**
- **Show HTML code**
- **Plead for domain NAME!**

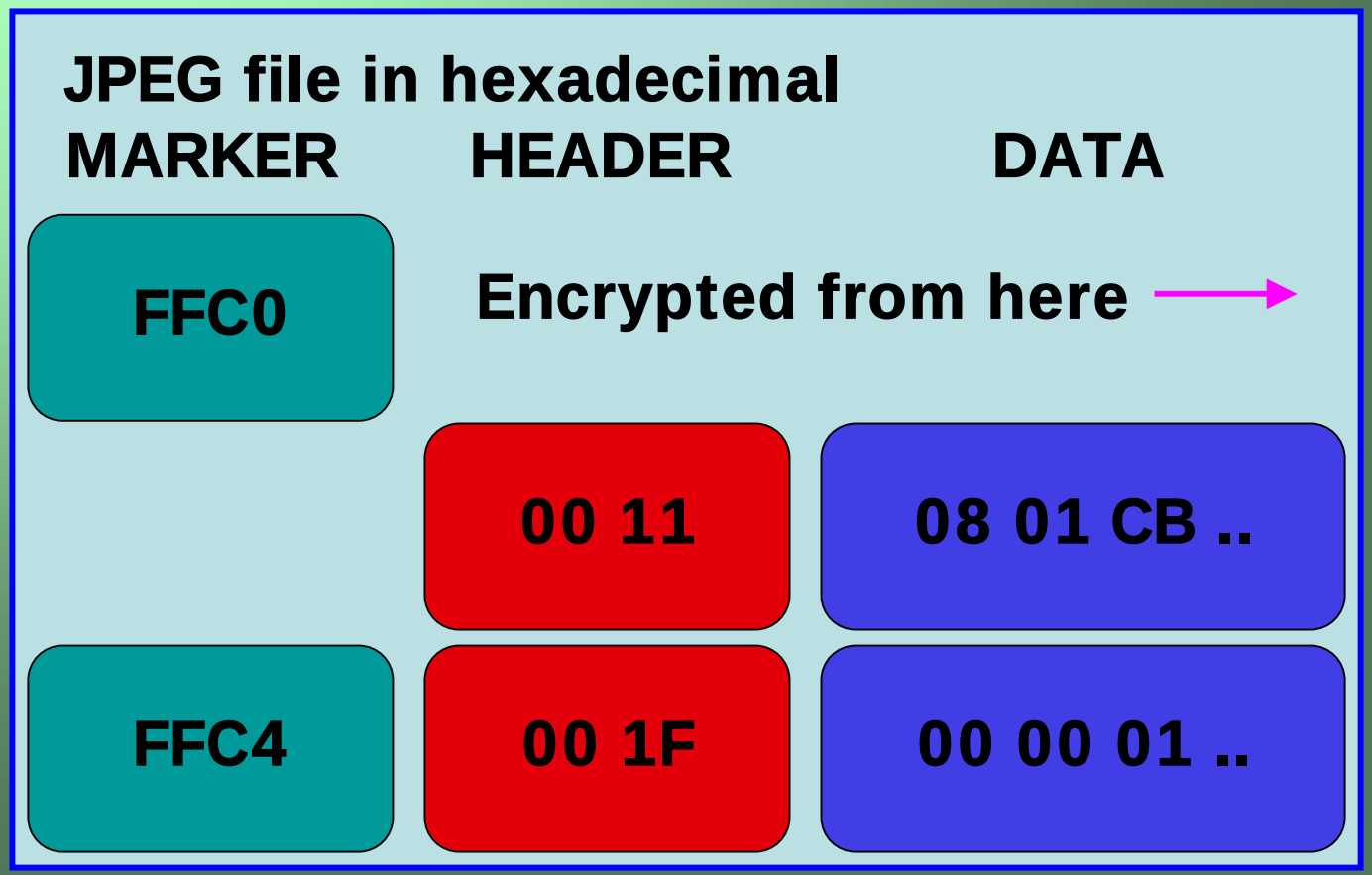
Presentation Overview:

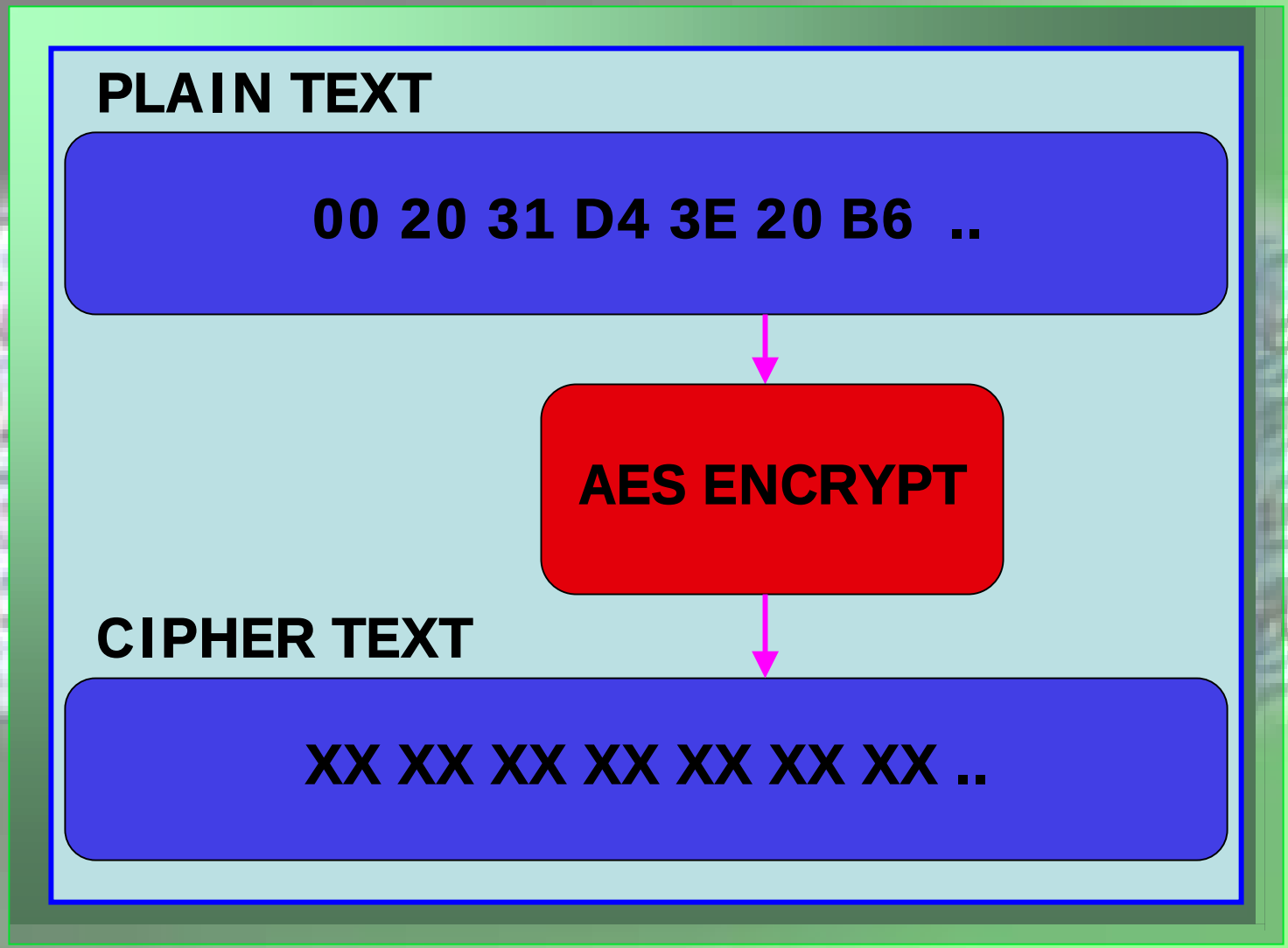
- Overview of Project
- Demonstration of
  - Manipulator
  - Production Code
  - Web Site
- Algorithm Design
- Potential Attacks
- Conclusion

Encryption Algorithm:

- **Write file-type-byte to ".ise" file**
  - **'1' for JPEG**
- **Read from input file**
- **Write unencrypted to ".ise" file**
- **Read/Write until Huffman**
  - **[FF C0 or FF C4]**
  - **baseline standard Huffman tables**

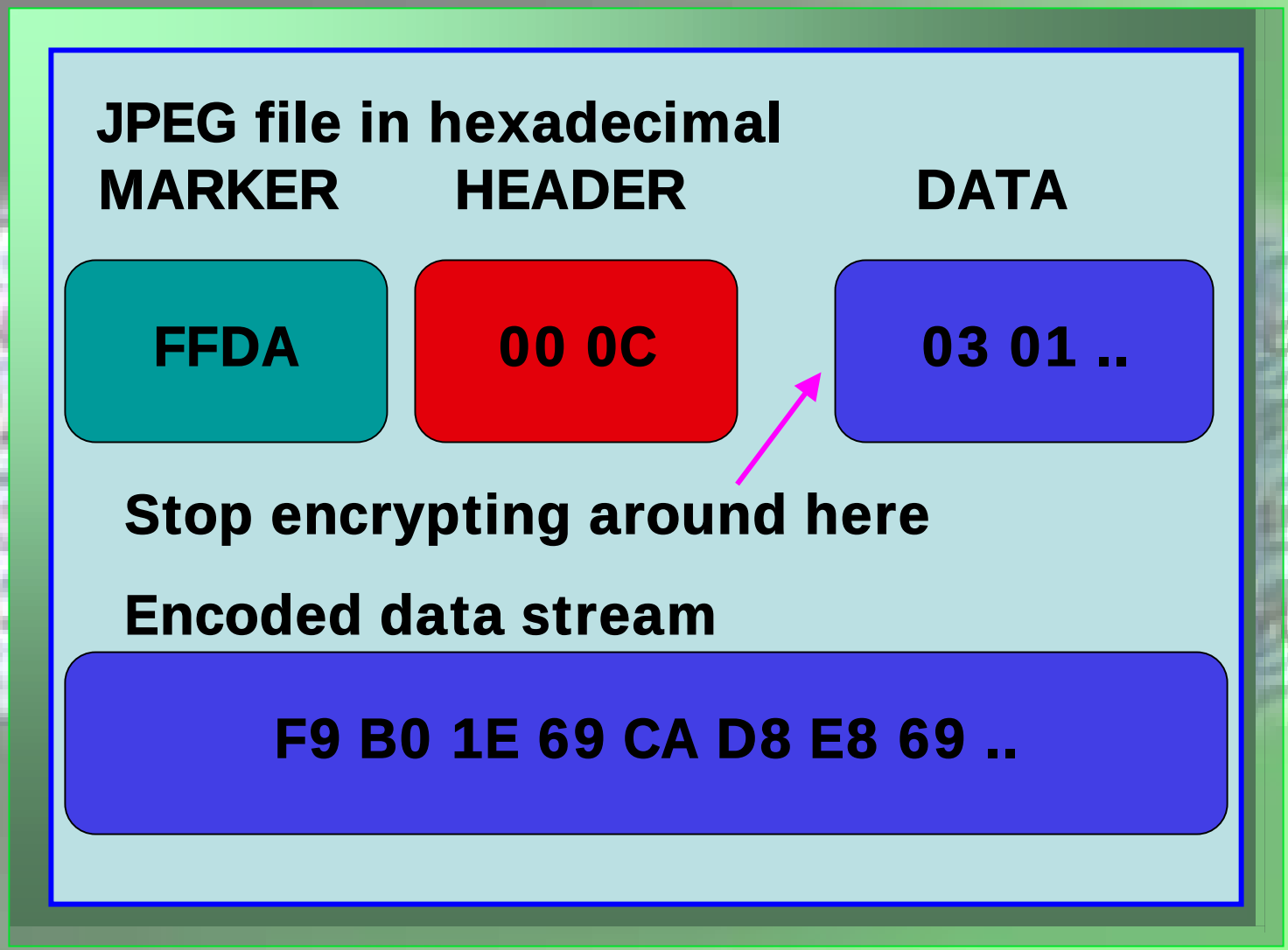
Start Encrypting After FFC0:




PLAIN TEXT
00 20 31 D4 3E 20 B6 ..
AES ENCRYPT
CIPHER TEXT
XX XX XX XX XX XX XX ..

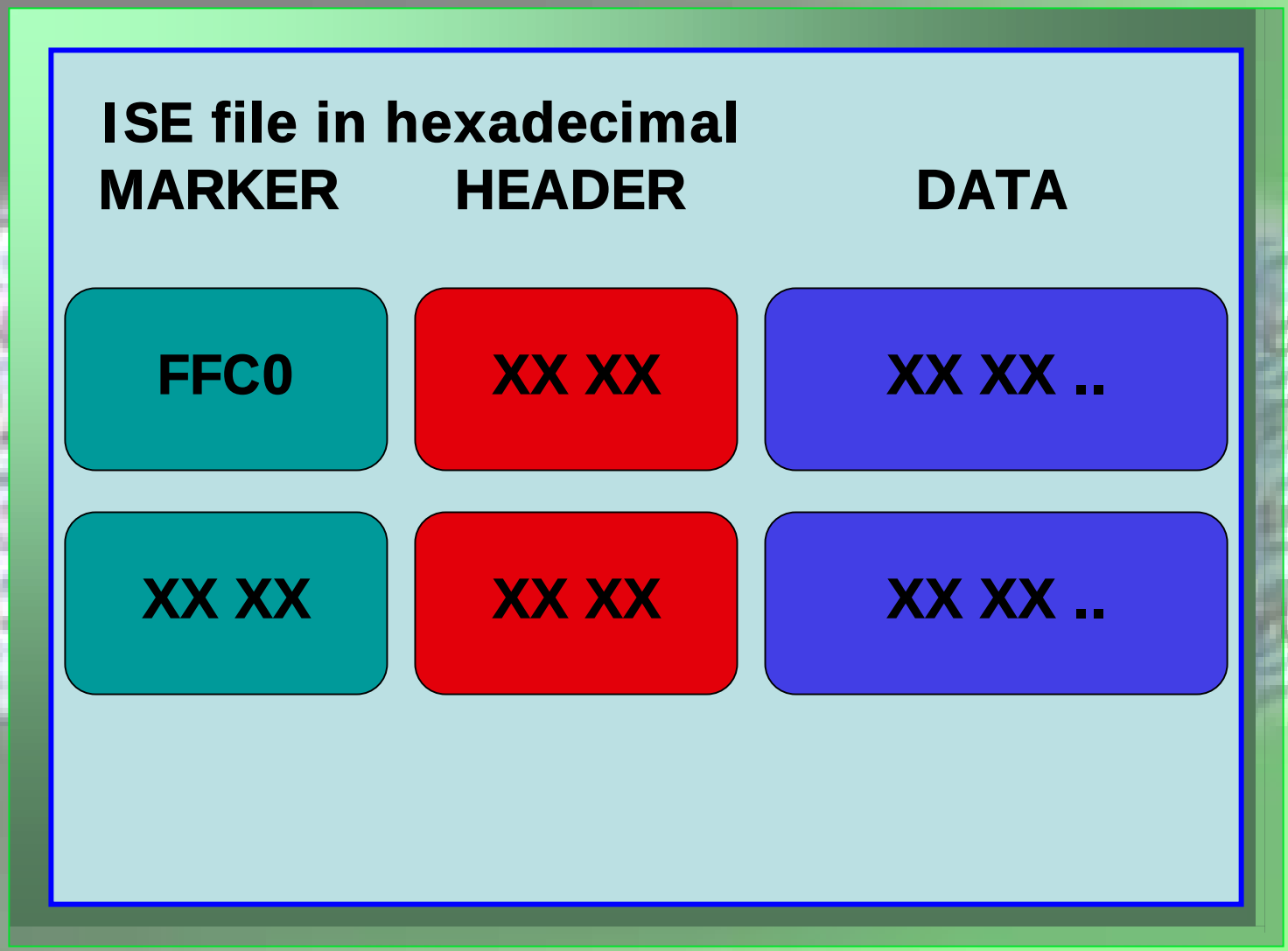
Encryption Algorithm:

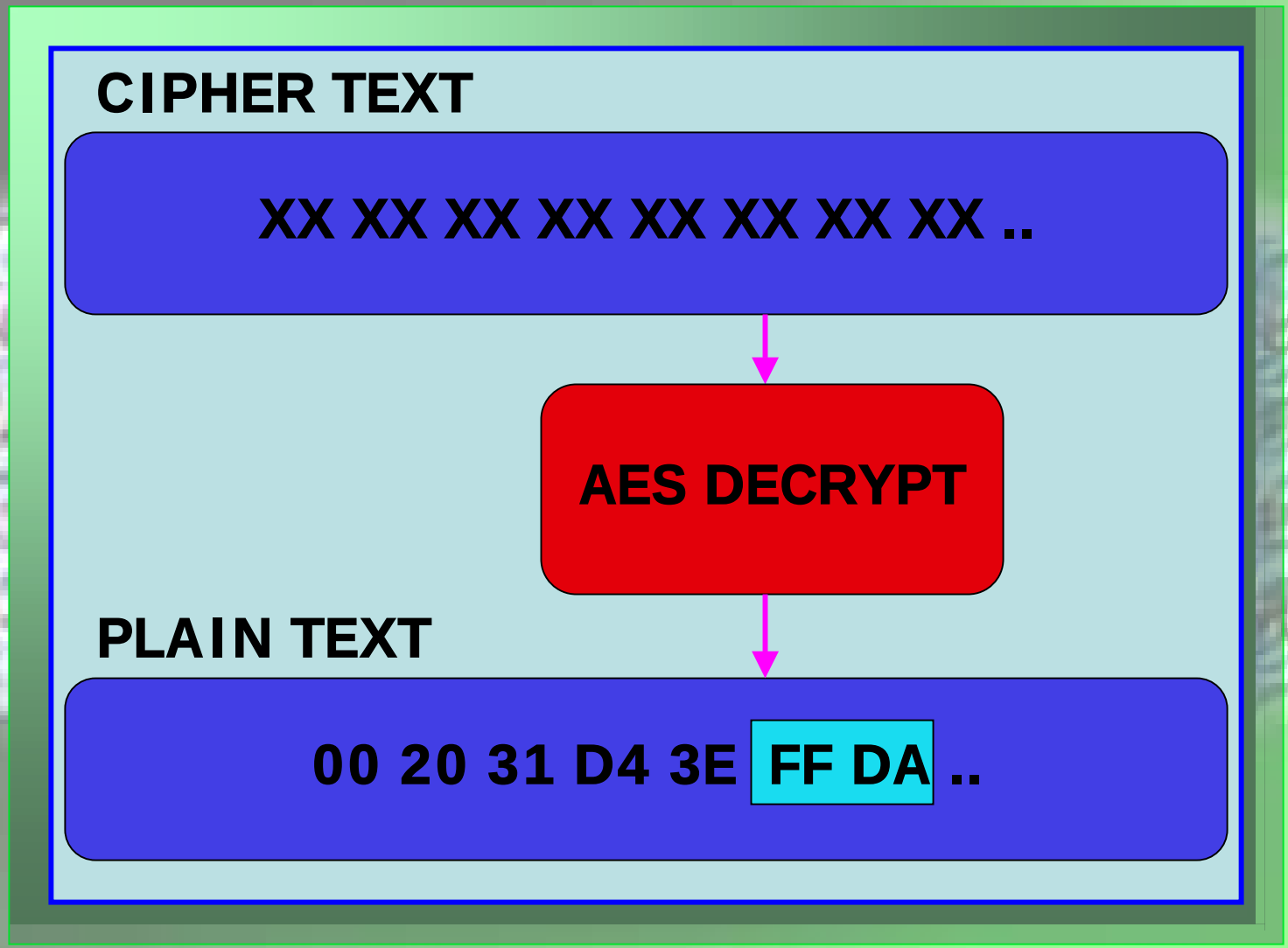
- **Keep encrypting until encoded data**
  - **[FF DA]**
  - **Start of encoded data stream**
- **Hide marker inside encrypted area**
- **Hide random length of encoded data**


JPEG file in hexadecimal
MARKER
HEADER
DATA
FFDA
00 0C
03 01 ..
Stop encrypting around here
Encoded data stream
F9 B0 1E 69 CA D8 E8 69 ..

Decryption Algorithm:

- **Read file-type-byte from ".ise" file**
  - **'1' for JPEG**
- **Read / Write until Huffman**
  - **[FF C0 or FFC4]**
  - **baseline standard Huffman tables**
- **Start decrypting**


ISE file in hexadecimal
MARKER
HEADER
DATA
FFC0
XX XX
XX XX ..
XX XX
XX XX
XX XX ..


CIPHER TEXT
XX XX XX XX XX XX XX XX ..
AES DECRYPT
PLAIN TEXT
00 20 31 D4 3E FF DA ..

Presentation Overview:

- Overview of Project
- Demonstration of
  - Manipulator
  - Production Code
  - Web Site
- Algorithm Design
- Potential Attacks
- Conclusion

Potential Attacks:

- **Brute force replacement**
- **Inside knowledge**
  - **Password**
  - **Image editor**
- **Could implement AES with larger**
  - **Key**
  - **Block length (further into data)**
- **Data is relatively untouched**
  - **Except at head**

Presentation Overview:

- Overview of Project
- Demonstration of
  - Manipulator
  - Production Code
  - Web Site
- Algorithm Design
- Potential Attacks
- Conclusion

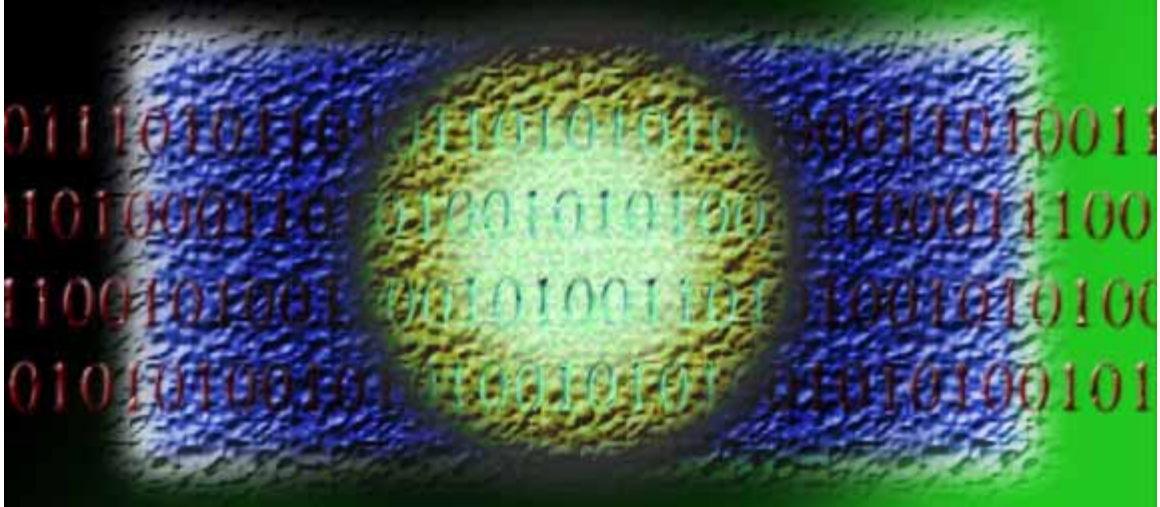
Questions

Developer's Reference

Team ISE

Image Selective Encryption

Developer's Reference
30 April 2004



Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Table of Contents

TEAM ISE DEVELOPER'S REFERENCE

Team ISE has deemed it necessary to supply those who will follow in our footsteps with answers to questions that will inevitably arise. Therefore, we have created a document broken down into three sections, one for each of the main products that we produced over the last year. The sections cover questions over the ISE Manipulator, Production Code, and Website. We also recommend viewing all of the documentation on our website. You can view to each section by clicking on one of the following links:

Also, Dr. Tom Lookabaugh at the University of Colorado at Boulder is responsible for maintaining all of the code and research produced by Team ISE. If you have questions or comments about any of the code, documentation or other items created by Team ISE, please contact Dr. Lookabaugh using the contact information given on the ISE Website. We hope you have as much fun and learn as much as we did – Good Luck!!!

Sincerely,

Team ISE
CSCI 4308 – 4318: Senior Project
University of Colorado at Boulder

ISE Manipulator

What is the ISE JPEG Manipulator?

The ISE JPEG Manipulator is a Windows application written in the C# (pronounced "Cee-Sharp") programming language and is design to allow the user to analyze and create JPEG images, based on the data from pre-existing JPEG images. The JPEG Manipulator was designed to help Team ISE with research during the course of the project. The Manipulator is extremely useful in determining how data (random or chosen) will react when integrated into a JPEG image. It is also a nice tool for analyzing the file structure of a JPEG image.

back

What does each of the code files do?

There are a total of four necessary code files included with the JPEG Manipulator. First, the most used code file is the frmMain.cs. This is the Main Form of the JPEG Manipulator and contains the frmMain form class and all of the code necessary to implement the base functionality of the Manipulator. The frmMain form class is directly inherited from the System.Windows.Forms class and provides the additional functionality as a series of methods in the class. Second, frmLoad.cs is the code file for the frmLoad form class inherited directly from the System.Windows.Forms class. This code was developed to provide a loading form to be displayed when a new image is loaded into the Manipulator. All of the functionality for this class is contained within the frmLoad.cs file and can be used as a separate .NET component. Third, the frmAbout.cs contains all of the code necessary to implement the frmAbout form class in the Manipulator. This class is also inherited directly form the System.Windows.Forms class. The fourth file is the frmSplash.cs code file. This file contains all of the necessary code to implement the frmSplash form class and is directly inherited from the System.Windows.Forms class. Please note that the latter 3 files are quite small, only a couple hundred of lines of code each, and only implement the functionality for the form's purpose.

back

Where can I find an in-depth design of the Manipulator?

There are several documents produced by Team ISE that explain the design for the ISE Manipulator. The ISE System Architecture document provides a nice high-level design of the ISE Manipulator. In addition, the ISE Design Specification provides an in-depth, high-level design as well as a description of each of the Methods necessary to fully develop the frmMain form. Also, for the convenience of future developers, the ISE Manipulator code, object and DLL files contain XML comments for each of the Methods

in the frmMain class, so that Visual Studio's Intellisense will display those comments (along with parameter details and return value information) in the comment viewer and in the Visual Studio Object Brower. This documentation is available at the website.
back

What do I need to compile the Manipulator code?

Although it is not required, Team ISE highly recommends working with Visual Studio .NET when creating and compiling this code. We have included a Visual Studio Solution file that contains all of the necessary file data to create, manage and build these codes very easily. This Solution file contains two Project files, one for the Manipulator itself and one for the Manipulator Installer package. But, since this is a .NET application, the only software truly necessary to build this program is the .NET framework. If the development machine does not have Visual Studio available, the developer always has the option of building the executable from calls to the command line compiler built into the .NET framework. For more information about using the command line compiler, please see www.microsoft.com/msdn. Again, Team ISE recommends using Visual Studio to manage and build these code files using the Solution file provided with the final distribution package.
back

What version of .NET was used for the Manipulator?

Version 1.1 of the .NET framework was used to develop the ISE Manipulator, but we believe it could be compiled with ANY version of the .NET framework (including the older 1.0 version). We have included the .NET framework re-distributable package for Windows with the final distribution package. Also, the .NET framework and can be attained by downloading it from Microsoft, if unavailable otherwise.
back

Back to top

ISE Production Code

What is the ISE Class?

The ISE Class is a C++ super-class, specifically designed to serve as a base for any selective encryption/decryption class targeted for compressed media. It contains all of the necessary methods and data members needed to construct an object for selective encryption, except for the actual encrypt and decrypt methods. Because the algorithms for selective encryption and decryption are uniquely tailored to a specific file type, these methods must be implemented in an inheriting class, such as the JPEG_ISE Class. When the JPEG_ISE class was implemented the code was written only to deal with standard baseline JPEG images.

back

How can I extend the ISE Class?

There are a number of steps required to inherit the ISE super class. First, one must design an algorithm to selectively encrypt a specified file type. These algorithms are then implemented within the inheriting class by defining the two virtual methods: encrypt_file() and decrypt_file(). All other protected methods within the super class are accessible by the inheriting class as needed.

back

Do I need to make my own constructor?

No, the ISE super class implements an overloaded constructor that should be used to construct any object inherited from ISE. **Note:** that the default constructor is declared protected and forces the class user to call the overloaded constructor. Also note that there are many ways to use the overloaded constructor. The only necessary parameter is the encryption/decryption key, but the user may also specify an input and output file.

back

Why are there only two files associated with an ISE object?

An ISE object can be used for either encryption or decryption. In each case, the only necessary data members are the input and an output file names. Setting these data members is dependent on the direction (encrypt or decrypt) of the object.

back

What methods does my inherited class have available to it?

There are a number of methods implemented in the super class. An inheriting class may use all of these methods. The methods available are:

- ise constructor
- set_key
- set_input_file_name
- set_output_file_name
- get_input_file_name
- get_output_file_name
- make_ise_file_name
- make_output_file_name
- get_ise_file_type

These functions allow for the manipulation of all the ISE object data members. A full description of each of these methods can be found at the associated link to the production code reference.

back

What methods will my inherited class need to implement?

There are only two methods an inheriting class needs to implement: encrypt file() and decrypt_file(). Both of these methods do not take any parameters and should use the data members of the object to selectively encrypt or decrypt a file. The algorithm for the encryption must be discovered through researching a given file type.

back

What changes to the ISE class should I make to implement my selective encryption class?

Aside from implementing the encrypt_file() and decrypt_file() functionality, there are minor changes that must be made to the super class. Every ISE file type must have a unique ID to be appended to the front of an ISE encrypted file in order to determine the original file type. JPEG ISE files have a ‘1’ as the ID. Also, the IDs ‘2’ and ‘3’ are reserved for mp3 and zip respectively, though their corresponding classes are yet to be defined. These changes need to be made in the ise get_ise_file_type() method and should be considered when creating the encrypt and decrypt functionality.

back

How do I design a selective encryption algorithm for my media file type?

There are a number of steps needed to design a new selective encryption algorithm. First, decide on which compressed media you would like encrypted and study its compression standard. Next, identify what portions of the file make good/bad targets for encryption. These targets should be evaluated for properties conducive to selective encryption. The factors to consider might be the portion of the file dedicated to the target, amount of damage to the file when the target is encrypted and how hard the encryption would be to break. The third step to designing a selective encryption algorithm is choosing the encryption. Team ISE used AES encryption, though other encryption algorithms may be used. The last step is to inherit the ISE super class and implement the encrypt_file() and decrypt_file() methods.

back

What encryption should I use in my algorithm?

Team ISE decided to use Rijndael AES encryption in the JPEG_ISE class because it does not increase the file size and the block cipher allows the randomization of number of bytes encrypted at the end of a section. Also, it came highly recommended by Professor Black of the University of Colorado at Boulder. However, if an encryption other than AES is well suited for your purpose, it can be used and will have no problems inheriting from the ISE Class.

back

Is there an example of a class inherited from the ISE Class?

Yes, Team ISE produced an inheriting class with the ISE production code. The JPEG_ISE Class is designed to implement selective encryption of standard baseline JPEG images. The class is freely available with the ISE super class.

back

Is there an example of how to use the JPEG_ISE Class?

Team ISE has included a main program using the functionality of the JPEG_ISE class. It was created to be an example of how the class could be used to selectively encrypt a JPEG image.

back

Where can I find more information about Team ISE and the ISE Production Code?

We have created a companion website that includes every research document we have created. This Website also contains links to many useful resources and even a bulletin board for discussions.

back

Back to top

ISE Website

How are the passwords obtained?
Where is the computer running the web page located?
What server is used?
What OS is the server running on?
What is the path to the directory with the web pages on the server?
Where are the images, documents, etc. located?
What is the IP address of the website?
Was an editor used to code the pages?
What version of HTML is used?
Are the pages written in valid HTML?
What character set are the pages coded in?
Do the pages make use of a CSS?
Do the pages follow a particular format?
What language was used to create the menu bar?
How can the menu bar be modified?
How is the menu bar included in the HTML pages?
Where was the message board obtained?
What is the message board coded in?
What type of database does the message board use?
Can the database be modified?
How can the database be copied?
How are the message board administrator options accessed?
How are new categories added?
How are categories deleted or edited?
How are new forums added?
How are forums edited?
How are forums deleted?
Can threads be automatically deleted?
Can the administrator manage the user accounts?
How are E-mail addresses banned?
How are E-mail addresses un-banned?
How are IP addresses banned?
How are IP addresses un-banned?
How are ranks set?
How are ranks deleted?

How are the passwords obtained?

Passwords are needed to log onto the computer the website is located on. Passwords are also needed to access the administrative options of the message board, and to log to the MySQL database used by the message board. These passwords were only provided to Dr. Tom Lookabaugh. In order to obtain these passwords, you must contact Dr. Lookabaugh.

back

Where is the computer running the web page located?

The web page is run off of a machine in Dr. Lookabaugh's Pervasive Computing Lab in the Discovery Learning Center at the University of Colorado at Boulder. To access this lab, you must obtain keys from Dr. Lookabaugh and the University of Colorado.

back

What server is used?

Apache/2.0.48 (Fedora) Server is used to serve this web page. This server comes with the Fedora Core Operating System, or can be downloaded from http://www.apache.org/. The particular version used to serve the web page was included in the Fedora Core release. The Apache server is configured to serve pages from a specific folder. The folder on the sponsor's computer is the "html" folder, which can be reached by following path from the root directory:

..\var\www\html

All pages that are to be served must be located in this folder.

back

What OS is the server running on?

The Operating System used by the server is the Red Hat Fedora Core Release. For more information on this operating system, please visit http://fedora.redhat.com/.

back

What is the path to the directory with the web pages on the server?

To reach the directory with the web pages, you must first log onto the computer where the server is located. You must log on as the root user. Obtain all passwords and usernames from Dr. Lookabaugh. Once logged on, the path to the directory with the pages is:

..\var\www\html

All of the pages are located in the "html" directory.

back

Where are the images, documents, etc. located?

The images, documents, and code accessible to the pages are located in subdirectories of the html directory. These subdirectories were created to organize all of the data used by the web pages.

All of the images are located in the "images" subdirectory. All images should be placed into this directory. Images can then be displayed on the web pages using the following HTML code:

```
<img src="images/ImageName">
```

ImageName refers to the actual name of the image plus its extension.

All of the documents are located in the "documents" subdirectory. All documents should be placed into this directory. Documents can then be called using the following HTML code:

```
<a href="documents/DocumentName"> LinkName</a>
```

DocumentName refers to the actual name of the document plus its extension. *LinkName* refers to the name of the desired link. It is recommended that all documents be in PDF format due to its portability.

All of the code is located in the "code" subdirectory. All downloadable code should be placed into this directory. Links to download the code can be created using the following HTML code:

```
<a href="code/CodeName">LinkName</a>
```

CodeName refers to the actual name of the code download plus its extension. *LinkName* refers to the name of the desired link.

All of the message board code is located in the "board" subdirectory.
The following path will lead to the menu bar code from the "html" directory:

images\menu

The images and JavaScript files used to create the menu bar are located in this directory.
back

What is the IP address of the website?

The IP address of the website and the computer it is located on is 128.138.75.184. The website is served off of port 80.
back

Was an editor used to code the pages?

Quantum Plus 3.1 was used to edit the HTML code of the web pages. It is recommended that this editor be used to modify the HTML code. It is not necessary to use this editor. However, the code was formatted using the spacing of this editor, and will be formatted properly in this editor.

back

What version of HTML is used?

The web pages are coded in HTML 4.01 Transitional. This allows some backwards compatibility with old HTML features. All pages associated with the website define the version of HTML being used by including the following header:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Any new pages added to this site should include this header and comply with HTML 4.01 Transitional standards.

back

Are the pages written in valid HTML?

The HTML used to code the pages is valid HTML 4.01 Transitional. The pages were validated using the HTML validator located at http://validator.w3.org. Any new pages should be validated using this validator. Additionally, any modifications made to the existing pages should be revalidated. To use the validator, visit the site located at the URL above and supply it with the address of the page that is to be validated. For example, if the download page is to be revalidated, type

http://128.138.75.184/DocumentIndex.html

into the address box on the validator website and click on the “Validate URI” button. The validator will return a report on any necessary fixes required to make the page valid.

back

What character set are the pages coded in?

The pages are coded in the ISO-8859-1 character encoding. All additional pages should include the following code within the header element to define the encoding:

```
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<META http-equiv="Content-Language" content="en-US">
```

These elements should be included inside of the <HEADER> element.

back

Do the pages make use of a CSS?

A cascading style sheet was not used to define the style of the web pages. The website was intended to be a very simple design that is only used to distribute code and documentation. Thus, no style sheet was defined, and the pages were constructed in a very simple manner.

back

Do the pages follow a particular format?

All of the pages have a header and footer. These should be included in any additional pages to ensure conformity to the rest of the website. The header and footer code can simply be cut and pasted into any additional pages. The HTML for the header is:

```
<TABLE bgcolor="#003300" align="center" width="100%">
 <tr>
    <td>
        <P>
            <FONT size="+1" color="#FFFFFF">
            <B>University of Colorado Computer Science Department</B><br>
            2003-2004<br>
            </FONT>
            <FONT size="+2" color="#FFFFFF">
            <B>ISE JPEG Selective Encryption Sponsor</B>
            </FONT>
        </P>
 </tr>
</TABLE>
<H1 align="center">
    <img src="images/ISE_icon_white.jpg" alt="Team ISE">
</H1>
```

This code should be included immediately after the <BODY> element.

The HTML for the footer is:

```
<P>
    <FONT size="2">
    This project was done by
    <a href="http://www.colorado.edu" target="_blank">University of Colorado</a>
    students under the supervision of the
    <a href="http://www.cs.colorado.edu" target="_blank">Computer Science Department</a>.
    </FONT>
</P>
<P>
    <FONT size="2">
    This Website is located on a sever at the University of Colorado at Boulder. Questions: Contact
    <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html" target="_blank">Tom
        Lookabaugh</a>
    or
    <a href="mailto:TeamISE@hotmail.com">TeamISE@hotmail.com</a>.
    </FONT>
</P>
```

```
<TABLE bgcolor="#003300" align="center" width="100%">
 <tr>
    <td>
        <P align="center">
            <FONT color="#FFFFFF">
                <B>Team Image Selective Encryption Sponsored by Tom Lookabaugh</B><br>
                <B>Department of Computer Science</B><br>
                <B>University of Colorado at Boulder</B><br>
                <B>Boulder, CO 80309-0430</B><br>
                <B>HTML 4.01 Transitional</B><br>
                <B>Copyright &copy 2003-2004</B>
            </FONT>
        </P>
 </tr>
</TABLE>
```

This code should be inserted immediately before the </BODY> element.
back

What language was used to create the menu bar?

The menu bar was created using the Xara Menu Maker tool. This tool generated the JavaScript used by the menu bar.
back

How can the menu bar be modified?

The JavaScript file is located using the following path from the "html" directory:

images\menu\isemenu.js

Xara Menu Maker is not required to modify this file. The file can be opened in an editor and modified to add additional menu or submenu items. There are two .js files in this directory. Only the isemenu.js file needs to be modified in order to add new menu items. The menu.js file should not be modified.
back

How is the menu bar included in the HTML pages?

The menu bar is included in the HTML pages using the following HTML code:

```
<SCRIPT src="images/menu/menu.js" type="text/JavaScript"></SCRIPT>
<SCRIPT src="images/menu/isemenu.js" type="text/JavaScript"></SCRIPT>
```

This code should be included immediately after the header.
back

Where was the message board obtained?

The message board was obtained from http://www.chipmunk-scripts.com/. This is a free software download that Team ISE made use of rather than creating their own message board.

back

What is the message board coded in?

The message board is coded in PHP with embedded MySQL queries. Because Team ISE did not develop this code, it should not be modified.

back

What type of database does the message board use?

The message board makes use of an underlying MySQL database. The installation script provided by the message board created all of the necessary tables utilized by the message board. This database software comes with the Red Hat Fedora Core release.

back

Can the database be modified?

The database can be modified by logging in as the root user and running the MySQL database. Refer to the password section to learn how to obtain the proper username and passwords. However, the database should not be modified to ensure the proper working of the message board. Additions to the message board can be made using the administrator options provided by the message board. This code will create the proper relations in the database needed for any changes. All modification to the database should be done indirectly through the administrator options.

back

How can the database be copied?

If the website is to be moved to another machine, the database tables must be copied from the MySQL server the database is currently in to the MySQL server on the new machine. If the new machine does not have a MySQL server, a server must be downloaded and installed by visiting the MySQL link. After the server is downloaded and installed, the database from the old server must be imported to the database on the new server. Refer to the MySQL online documentation for instructions on coping databases to another machine.

back

How are the message board administrator options accessed?

First, the administrator login name and password must be acquired. Refer to the section on passwords for information on how to obtain these items. Once these items are known, click on the "Login" link in the upper right corner of the message board page directly under the header. Provide the text boxes with the correct administrator username and password. The message board will then redirect you to the forum page. Centered at the bottom of the forum page is a link called "Admin CP". Clicking on this link will open a

window in which all of the administrator options can be accessed by clicking on the links. Click on one of the options under the "Admin Options" table to use the option.
back

How are new categories added?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Add category" link. Provide a name for the category and its rank. The category's rank determines the order in which the category is listed in relation to the other categories. The lowest number category is listed first. After entering this information click on the "submit" button. The message board now contains a new category.
back

How are categories deleted or edited?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Delete/Edit Category" link. All existing categories will now be listed in a table. The category name will be listed in the left column, a "Delete?" link in the middle column, and an "Edit?" link in the right column of the table. To delete a category, click on the "Delete?" link found in the middle column of the table. The board will ask you if you are sure you want to delete the category. Click on the "Main" link on the "Admin Options" table to cancel the delete. To edit the category, click on the "Edit?" link in the right column of the table. You will be redirected to the add category window. However, the category name and rank are already filled in. Make desired changes to the name and rank and click submit. The edited changes will now be applied.
back

How are new forums added?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Add Forum(s)" link. This will redirect you to the add forum window. Once in this window, you can create a forum by filling in a name in the "name of forum" text box. Next, choose which category to associate the forum with by clicking on the pull down menu. The menu will be populated with all of the categories on the message board. Select the category and then set the permission level to read, post, and reply in the forum you are creating. Selecting one of the permissions from the pull down menu for each read, post, and reply can set the respective permission level. In the last text box, type a definition of the forum. After all text boxes are filled and all permissions set, click on the "Create Forum" button to create a new forum with the information you inserted.
back

How are forums edited?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Edit Forums(s)" link. A table will be displayed listing all of the different forums for each category. Click on the "Edit" link next to the forum you desire to change. This will open the same window as when clicking on the "Add Forum(s)" link, except all of the information will be filled in. Change any desired information and click on the "submit" button to make the changes take effect.

back

How are forums deleted?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Delete Forums(s)" link. A table will be displayed with all of the forums available for the different categories. The name of each forum will be displayed in the left column of the table, and a "Delete this Forum" link can be found in the right column of the table. Click on the "Delete this Forum" link next to the forum you desire to delete. The board will ask you again if you want to delete the forum. Click on the "Delete this Forum" button again to delete the forum. Click on the "Main" link under the "Admin Options" table to cancel the deletion.

back

Can threads be automatically deleted?

Threads without responses can be automatically deleted after a certain number of days. To adjust the amount of days, first log on as the administrator, click on the "Admin CP" link and then click on the "Prune Topics" link. The administrator can then enter the amount of days before a thread is to be deleted. If the thread does not receive a response after the allotted time has passed, it will automatically be deleted.

back

Can the administrator manage the user accounts?

The administrator can manage user accounts. After logging in as the administrator and clicking on the "Admin CP" link, click on the "User Management" link. Type the user name into the text box and click on the "Search for User" button. If you want all of the users to be displayed, leave the text box empty and click on the "Search for User" button. After clicking the button, the username(s) will be displayed along with the user's email address. To edit a user's profile click on the "Edit" link next to the user's name. The administrator can then edit the user's profile by changing the information in the text box fields. Clicking on the "Edit User" button will make the changes permanent. The administrator can also delete a user by clicking on the "Delete" link next to the user's name. The message board will prompt the administrator again before making the deletion permanent. Clicking on the "Delete This User" button will complete the deletion. To cancel the deletion, click on the "Main" link under the "Admin Options" table.

back

How are E-mail addresses banned?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Ban E-mail" link. The administrator can then fill in the text box with the E-mail address that is to be banned. Click on the submit button to make the ban take effect. Any users with a banned E-mail address cannot register to use the message board. **Note:** If the text box is filled with a generic E-mail, such as "@hotmail.com" all users will an E-mail address ending with "@hotmail.com" will not be allowed to register on the message board.

back

How are E-mail addresses un-banned?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Unban E-mail" link. A list of banned E-mail addresses will be displayed. Click on the "Delete" button next to an E-mail address to un-ban that address. Users with that E-mail address will now be allowed to register to use the message board.
back

How are IP addresses banned?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Ban IP" link. The administrator can then fill in the text box with the IP address that is to be banned. Click on the submit button to make the ban take effect. Any users trying to register from a banned IP address will not be able to use the message board.
back

How are IP addresses un-banned?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Unban IP" link. A list of banned IP addresses will be displayed. Click on the "Delete" button next to an IP address to un-ban that address. Users working from that IP address will now be allowed to register to and use the message board.
back

How are ranks set?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Add Rank" link. The number of posts the user has made determines the rank of a user. To create a rank, the administrator creates a rank title by filling in the "Rank" text box with the title name. The administrator then fills in the "Number of Posts Needed to Achieve" text box with a number. Clicking the "submit" button will create this rank. A user will automatically be assigned the rank once he or she has submitted enough posts.
back

How are ranks deleted?

After logging in as the administrator and clicking on the "Admin CP" link, click on the "Delete Rank" link. A list of ranks will then be displayed along with the number of posts needed to achieve such a rank. Clicking on the "Delete" link next to the rank will allow the administrator to delete the rank. The message board will then prompt the administrator again about deleting the rank. Clicking on the "Delete Rank" button will delete the rank. To cancel the deletion, click on the "Main" button under the "Admin Options" table.
back

Back to top

Conclusion

This Developer's Reference is intended for anyone who plans to modify or extend any of the ISE products or to continue with the research begun by Team ISE. The reference is provided as a supplement to the other design related documents provided in the ISE release. The intent of this document is to answer any questions about modifying or extending any of the Team ISE products or continuing with the ISE research. For further clarification, please refer to the Requirements document, the System Architecture document, the Design document and the reference manuals of the products.

Back to top

Research Paper

Team ISE

Image Selective Encryption

Selective Encryption of JPEG Standard Baseline Compression Images

Tom Lookabaugh, Andrew T. Pouzeshi, Geoffrey L. Griffith, Joe B. Jarchow, Joseph Z. Kadhim, Shinya Daigaku
Department of Computer Science, University of Colorado, Campus Box 530, Boulder, CO, USA 80309-0530

Abstract

One of the ramifications of compressing a file is that vital data are localized in small, specific areas. Consequently, it is easy to exploit this property of compression to provide a high level of security as selective encryption focuses on encrypting only these vital portions of data to render a file unusable. Selective encryption results in a large savings in computationally intensive operations, while maintaining a reliable level of security. There have been a number of selective encryption methods proposed for the JPEG compressed image format. This paper describes a simple, yet secure method for selectively encrypting JPEG images that are compressed using the Baseline[1] standard. JPEG Selective Encryption has a high value for any application in which sensitive images may be at risk, from low power satellite imaging systems to securely transmitting images across the Internet.

Keywords: JPEG, image encryption, encryption, selective encryption, partial encryption, cryptography, cryptanalysis, compression, security.

1. Introduction

Selective Encryption is defined as applying encryption to a portion of a file's bit-stream with the assumption that the entire file will become useless without the proper decryptor. The attractiveness of selective encryption arises from the idea that a file can be securely encrypted and transmitted without spending the computational effort of encrypting the entire file. Selective encryption techniques range from encrypting a portion of the file, say a straight percentage of the data, to others that encrypt specific vital sections of a file. Selective encryption methods are never as secure as encrypting an entire file, because much of the data is not encrypted. The goal of selective encryption is to reduce the computational time of encryption, while maintaining a sufficient level of file protection.

The increase of multimedia applications and the transmission of data over public networks necessitate efficient methods of securing transmitted data. Because of the large size of multimedia files, Selective Encryption methods have been devised for various different types of multimedia compressions. The increasing of use of JPEG image encoding software and hardware and transmission across large public networks warrants a strong, yet simple, Selective Encryption scheme for JPEG images. The goal of the research behind this paper was the development of a simple yet secure method of Selective Encryption for the JPEG Baseline compression standard. Such a method would be applicable in situations ranging from encrypting transmitted satellite imagery to encrypting images generated by digital cameras to protecting images for transmission across the Internet.

---

[1] As defined by Pennebaker and Mitchell in "JPEG Still Image Data Compression Standard."

2. JPEG Image Selective Encryption Criteria

The goal of Selective Encryption for JPEG images is to minimize the amount of encryption applied to a file while maximizing the damage done to the image. As a bonus, this paper will define a method that is relatively fixed in size and will not require the amount of encryption to increase linearly as the image size increases linearly. Most of the file will remain unencrypted, which allows the retrieval of those data. However, the data that remains unencrypted will be useless without the encrypted data and the encrypted data will be reasonably difficult for a hacker to replace, reconstruct or calculate. The result will be a JPEG file structure, partially encrypted, that is impossible to use without the proper decryptor and key. The focus of this paper is to present the research and algorithm developed for selective encryption of standard Baseline compressed JPEG image files.

3. Goals and Criteria for Selective Encryption

Selective encryption can be measured in several different ways and optimized for many different purposes. Confusion may arise from reading literature about selective encryption, as the method is usually specific to the type of file being encrypted. Thus, this confusion can be avoided by having a clear idea of our selective encryption criteria. The criteria used for selective encryption include:

Security Criterion:

Selective encryption has been proposed for a number of different user scenarios. For the purposes of this paper, we will define the security criterion as encryption of data sufficient to render the image unusable to a standard JPEG image decoder. The data must be vital enough to render the image unusable to an attacker's reconstruction or replacement of data to the point that the attacker would be forced to use an expensive brute force method to decode the image. Although the attacker will still be able to retrieve most of the data from the selectively encrypted image file, the image itself cannot be easily reconstructed.

Security Validation:

Security of any given encryption can be validated in a number of different ways. Some researchers validate security by choosing the criteria and then feeding the selectively encrypted data into a standard decoder and observing resulting reconstructions. Other researchers take a cryptanalytic approach by acting as an attacker and working with a modified decoder and other available information to design a method of defeating the selective encryption. Still others make mathematical calculations, such as RMS (root mean squared) or PSNR (peak signal-to-noise ratio), to find differences between the encrypted and unencrypted data values. This paper considers all three of these methods as valid and all three have been considered for this cryptosystem.

Complexity:

Often encryption can be complex and computationally expensive. The primary goal of selective encryption is to reduce the percentage of data that needs to be encrypted, while maintaining an acceptable level of security. This reduction of encryption operations must

be weighed against increased operations necessary to implement the selective encryption algorithm. If computing the data to encrypt and/or searching for those data is more expensive than simply encrypting the entire file, then the selective encryption system should not be considered as valid.

Compression Efficiency:

The primary goal of compression is to store a set of data in less space than the data representation requires by utilizing specialized algorithms. Image, video, and audio compression formats often exploit the fact that a human detects only a portion of the overall sensory input. Therefore, certain compression formats further reduce the number of bits needed to represent the data by approximating the data values such that the difference is relatively undetectable to human sense. Note that in these cases, the exact data are lost, but the compression efficiency will be much greater. However, the quality of the media is degraded exponentially by increasing the overall compression. For this reason, compressors often allow for varying degrees of data loss.

Some methods of selective encryption compromise compression efficiency by adding data overhead and/or by modifying the compression algorithm, causing a penalty in performance. For example, constantly searching a data stream for information about where to encrypt will add computational overhead. In addition, certain encryption algorithms greatly increase the size of the data, which is contrary to compression. Although there are newer encryptions that do not increase data size, circumstances may require use of less size efficient encryption algorithms. Any degradation in compression efficiency must be weighed against the constraints surrounding the particular need for selective encryption.

Interaction with Compressors:

There are methods of selective encryption that work with, and others that are independent, of the compression algorithm. It is important to be aware that there are potentially major differences in both performance and compression efficiency between these two methods. Ideally, the selective encryption algorithm would be implemented within the compression algorithm. This will minimize file parsing operations and reduce the overall number of operations needed to find the portion of data to encrypt.

Selective Encryption Attacks:

One final item to define and consider are the types of attacks on a selective encryption system. There is a clear difference between cracking a particular selective encryption system and cracking an encryption algorithm. If the encryption algorithm used to implement the selective encryption system is breakable, or found to be breakable in the future, we must assume the selective encryption system is invalidated and we must switch to a more secure encryption algorithm. For the purpose of this paper, we will assume the particular encryption algorithm used to implement JPEG selective encryption system is secure and we will discuss only attacks that pertain to selective encryption and not the various attacks on different encryption algorithms.

There are really three ways of attacking a selective encryption system. The first, and most well known, is a purely brute force attack. Since selective encryption systems only encrypt a portion of the file, usually the minimum possible to sufficiently protect the data, it will take much less time to either defeat the encrypted data or remove the encrypted data and try a systematic bit replacement of all possible values. For JPEG selective encryption, an attacker would work with a standard JPEG image decoder and would run each permutation of the replaced data through the decoder to try to rule out the most obvious, unviewable images. Then, once the automated process has produced a number of images that are viewable, the attacker would have to look at each one to find the correct, or at least understandable image. However, this will potentially take a large amount of time, depending on how many bits are encrypted. So, we will define the brute force attack as the most undesirable method.

The second method of attack is defined as a reconstruction attack. An expert in the particular file format that is selectively encrypted could devise a method for reconstructing the vital data that have been encrypted, given the unencrypted data in the file. In the case of JPEG selective encryption, the attacker would work with a modified JPEG image decoder that would compute the correct information, given the selectively encrypted file. Fortunately, the JPEG compression standards (along with many other compression standards) are designed specifically to decompose the original image into its vital components that allow the decoder to calculate each pixel value to reduce overall file size. Ironically, it is this reason that makes selective encryption of compression formats very attractive. Still, and in general, it is important to be extremely familiar with the data format of the selectively encrypted file, and measures must be taken to avoid this form of attack.

Finally, the third, and probably most effective, would be a hybrid attack. There are several possibilities for this type of attack, which would consist of doing research on real world instances of the particular file format to try to find consistency of vital data and having some understanding of how the data are structured. This could help the attacker by reducing the amount of "most likely" possibilities of that data. For JPEG selective encryption, this would probably entail a basic understanding of the JPEG image data components and some prior knowledge of a large number of real world instances of JPEG images and commonly used JPEG encoding schemes. Then, it could be determined if the encrypted data could be replaced by trial and error with a relatively small number of sets of real world data to try to reproduce (or even approximate) the original image to an acceptable level. Again, measures must be taken to ensure this type of attack will lead to failure.

4. Previous Selective Encryption Attempts

There is currently a small amount of existing research on the topic of Selective Encryption available for various multimedia formats. Much of the research pertinent to this paper is based on previous MPEG and JPEG Selective Encryption techniques and research. Through out this literature there is much indecision as to which file components are the best target(s) for Selective Encryption. This paper attempts to

evaluate all possible targets and previous attempts of Selective Encryption for JPEG image formats and any possible attacker's counter measures.

In their paper "Selective Encryption of the JPEG2000 Bitstream," Norcen and Uhl [8] outline a selective encryption method for the JPEG2000 compressed file format. The proposed method uses an AES block cipher to encrypt 20% of the visual information in JPEG2000 files, providing relatively secure file transmission without the computational costs of encrypting the entire file. While this method is more efficient than encrypting the entire file, the algorithm fails to exploit the relationship between compression and isolation of vital data. Moreover, the amount of data that is encrypted in the file increases linearly as the JPEG file size increases. Ideally, the amount of data encrypted would be relatively fixed and would include only vital components that would render the image unusable.

By carefully selecting vital components of the file to encrypt, it is possible to provide security while encrypting an even smaller, and ideally fixed, portion of the file. Several other research papers (mostly concerning MPEG Selective Encryption) suggest targeting the DCT (Discrete Cosine Transform) Quantizer tables found in many compressed multimedia file formats, including JPEG formats. The DCT is a mathematical technique used for decomposing wavelengths into elementary frequency components. For a JPEG image, these coefficients are stored in the Quantizer table. Encrypting the Quantizer tables are an attractive target because there is no variance in table size and the number of tables allowed is small, yet the minimum amount of Quantizer data is not so small that it could easily be permuted or guessed. Each Quantizer table must be exactly 64 bytes, and there are no less than 1 and no more than 4 allowed. Thus, there is a minimum of 2^{512} possibilities and up to 2^{2048} possibilities to guess the exact Quantizer table(s) encrypted. This target is also large enough that a non-intelligent brute force attack of simply substituting values for these tables would take a considerable amount of time to reproduce the original image. Even though the Quantizer table looks promising at first glance, it proves to be an extremely weak target for JPEG selective encryption, as we'll see in section 5 of this paper.

In their paper "Secure Compression using Adaptive Huffman Encoding," Kailasanathan, Naini, and Ogunbona [4] propose the Huffman encoding tables, found in the Baseline JPEG format, as a viable target for selective encryption. This selective encryption algorithm offers two possible solutions. The first involves removing the compression tables from the image, securely transmitting the tables separately, and then reintegrating the tables when received. The second, more appealing solution, is to encrypt the compression table and send it along with the file and then securely transmitting a key to decrypt on the other side. As with the Quantizer tables, the Huffman tables appear to be a good target for selective encryption, because these tables have a relatively small variance in size, yet the minimum size is sufficient to repel brute force attacks. After further research discussed in section 5 of this paper, the Huffman tables prove to be a more valuable a target for selective encryption. Unlike the Quantizer table values, it is not as easy to produce an image by replacing the Huffman values of an optimized JPEG image. However, because many JPEG compression applications use default Huffman

tables, an attacker may have success by trying a series of popular default tables used by the more common graphical editing applications, digital cameras (JPEG encoding chips) or the example tables in the JPEG standard. Still, both Quantizer and Huffman seemed to have potential, and in the end, the research finally yielded a solid solution.

5. Cryptanalytic Approach to JPEG Selective Encryption

To devise an algorithm for selectively encrypting JPEG images effectively, the team researched the feasibility of this project from several different angles. Since there is no universal method for selective encryption, the team thought it appropriate to examine previous research on subject for multiple compression formats, review the JPEG baseline compression standard[2], research of common implementations of the JPEG encoders/decoders, and collect a large sample of real world JPEG images to be used for statistical analysis. By the end, the team was able to devise a method of selective encryption that will sufficiently protect JPEG images against any of the possible attacks mentioned in this paper.

The team began by researching the Baseline standard compression for JPEG images. Although there is a large amount of data included in the format, much of it is not vital to the image, or can be replaced, or even calculated. The team narrowed the possible targets for selective encryption to three pieces: the Encoded Data stream, the Quantizer tables, and the Huffman tables (which coincided with previous research available).

As mentioned above, a previous attempt at the Selective Encryption of JPEG images was to encrypt a percentage of the entire Encoded Data. While this method will definitely work, it was ruled out for two reasons. The first, and the most important reason, is that non-intelligently encrypting a percentage of the Encoded Data fails to exploit the relationship between a compression format and the concentration of vital data. Second, the amount of encryption needed will linearly increase as the size of the file increases. The Encoded Data makes up the largest percentage of the file size (on the order of 96% for JPEG images under 20 KB and 99%+ for files of 200 KB or more). The goal is to have a relatively fixed amount of data that needs to be encrypted and ideally that size will not be dependent upon the image size. Thus, the Encoded Data was ruled out as a viable target.

Another possible target found from both analyzing the JPEG standard and reviewing previous research is the Quantizer tables. There was a considerable amount of selective encryption research available for methods that utilize the Quantizer tables, but much of it was for other compression formats. However, there were at least two research papers on the topic of selective encryption for JPEG images that suggested the Quantizer tables are good targets. With this in mind, the team decided to try working with this Quantizer to see what effect, if any, altering these values had on various images. During the course of the research, over 2500 random JPEG images were gathered from the Internet and over 200 were tested directly. Unfortunately, it was determined that this target was neither vital enough nor unique enough to provide ample security. Altering the DCT coefficients only distorts the resolution, brightness, or color. Even a completely random table would

[2] As defined by Pennebaker and Mitchell in "JPEG Still Image Data Compression Standard."

yield a viewable image of the original only slightly degraded. In many cases, the team was able to reconstruct most images by simply replacing the entire table with a single value for each of the DCT coefficients, allowing the image to decode with a negligible degradation of quality. Although the images were often slightly discolored and/or the resolution was distorted, these images were certainly not damaged enough to render them incomprehensible. For this reason, the Quantizer tables were ruled out as a viable target.

Finally, the team focused on the Huffman (compression) tables as a target for selective encryption. The image was found to be extremely sensitive to minor changes in the Huffman tables, as these tables are used to generate/decode the Encoded Data stream. If even one encoding value is altered, then the resulting image will be considerably damaged. Furthermore, it will be impossible to reconstruct images by replacing Huffman tables with random values or even different Huffman tables from other images. Unlike the encoded data stream, the size of these tables is relatively fixed, as the Baseline standard dictates that there can be a maximum of four of these tables. So, on the surface, and as other research pointed out, the Huffman tables seem to be the most attractive target for JPEG selective encryption. However, it is necessary to look more into JPEG compressors and common instances of JPEG images to validate the security of a selective encryption method that targets the Huffman tables.

There are a wide variety of different JPEG encoders available, such as the IJG[3] JPEG encoding/decoding classes, Adobe Photoshop (a professional image editing application) or even the common Microsoft Paint (included with every copy of Microsoft Windows). While each encoder provides a different level of features, they all work with the JPEG Baseline compression standard. The main differences among these encoders can be measured by how they actually encode the image itself. While some encoders will actually calculate an optimized Huffman table, others use a series of default tables that are pre-calculated. Although these pre-calculated tables reduce computation, they pose a problem to security, because if an attacker had "inside information" on which JPEG encoder was used, they might be able replace the encrypted compression table. Due to the existence of default compression tables, a selective encryption method that only encrypted the Huffman tables would be insecure.

A remedy to solve the problem with default Huffman tables would be to optimize the compression of every JPEG image, before selectively encrypting. However, there are two potential problems with this remedy. First, using the IJG compressor with a flag to optimize images, the team produced approximately 470 optimized JPEG images. These images were randomly collected from the Internet. Even after optimization, there were still a large number of duplicate Huffman tables. Of these non-optimized images, 76.3% contained duplicate Huffman data. After optimizing these same images, 39.6% contained duplicate Huffman data. Thus, even after optimization, a considerable number of duplicate tables still existed, meaning that even if images are optimized, attackers may still be able to replace these values. Secondly, a goal of selective encryption is to reduce the amount of computation necessary to protect the file. However, by optimizing JPEG

[3] The IJG Organization is one of the most common providers of a C++ API for encoding and decoding JPEG images.

images (i.e. not making use of pre-calculated tables), there is an increase in the amount of computation needed to assure security of the image. Moreover, many of the JPEG compression chips used in digital cameras or satellite systems do not have the capability of calculating an optimized table. So, although the Huffman tables seem like the perfect target, they alone do not provide the level of security selective encryption hopes to achieve.

After spending a considerable amount of time researching, it became increasingly apparent that just encrypting one or two frames of data in the image wasn't going to solve all of the problems. The attacker could know at least the size of the table and the number of tables for both the Quantizer and the Huffman tables by counting encrypted frames in the image. Moreover, the Huffman tables have an ordering which greatly reduces the number of possible permutations and the Quantizer tables by themselves are much too weak because even a randomized table will often produce a degraded image, but not damaged enough to make it completely unusable. The team realized that we needed to hide the exact size and number of the compression tables.

To overcome all of these drawbacks, Team ISE devised an algorithm that encrypts not only the compression data frames, but also all the data between the compression tables and the beginning of the Encoded Data stream. The Team ISE algorithm can be implemented in cooperation with compression or independent of compression, as well as in software or in hardware. The algorithm is as follows:

1. Choose a block size of some number of bytes (for example, 32 bytes work well with the AES block cipher encryption system).
2. Write the file as normal until the FFC0 (SOF0 frame) or FFC4 (DHT frame) marker (whichever is written first for the particular encoder).
3. Write this 2 byte marker and then start encrypting in blocks of the pre-chosen block size until the FFDA marker (SOS frame) is to be written.
4. Encrypt the FFDA marker and fill out the rest of the current block and write it to file.
5. Encrypt one final block and write it to file.
6. Write the rest of the Encoded Data stream and file as normal.

This effectively hides the size of the Huffman tables within the file. This causes the encryption to run directly into the Encoded Data stream. Since both the encryption and the encoded data stream appear to be random values, it is now impossible to tell where the Huffman tables end and the Encoded Data begins. Thus we have overcome the problem of direct table replacement. Furthermore, a brute force attack would be extremely expensive, because the average size of these tables for a small image would yield about 2^{2400} possibilities! This leaves only the problem of the Hybrid attack with (1) "inside information" of a compressor that (2) uses pre-calculated or default compression tables that are unchanging. In this case, an attacker could replace the encrypted table and recalculation of the Scan header frame. Any data that was encrypted at the beginning of the Encoded Data stream could be systematically substituted until the correct solution is found. At a minimum, the Hybrid attack method would have (assuming a 32 byte block

size) at least 2^{256} possibilities and at most, there would be at most 2^{512} possibilities. Thus, this particular Hybrid attack would still be very expensive and take quite a bit of time and effort by the attacker. However, the key to overcoming this attack is to use an optimized compression algorithm for the table. Moreover, this cryptosystem encrypts only about 3% of the JPEG image data for a very small image around 20 KB and for the case of a image produced by a digital camera (of about 1 MB in size), this selective encryption algorithm will encrypt only about 0.001% of the file.

6. Conclusion

After researching previous attempts at JPEG selective encryption, we found that although previous researchers were definitely on the right track, there are many weaknesses in the other approaches. The algorithm developed by Team ISE overcomes these weaknesses while adhering to the original goals of selective encryption defined in this paper. The algorithm performs in such way that the number of computational operations needed to encrypt the data does not increase as file size increases. Furthermore, the algorithm is simple enough that it can be easily implemented in both software and hardware, in cooperation or independent of the compressor, thereby lending itself to provide high flexibility for many different applications. The Team ISE selective encryption algorithm will only be vulnerable to a brute force attack. The algorithm defined here has met all of the goals set out in this paper and finally, but most importantly, the algorithm is secure.

Bibliography

1. Chang, H. and Li, X. *On the Application of Image Decomposition to Image Compression and Encryption*. 1996.

2. Chang, H. and Li, X. *Partial Encryption of Compressed Images and Videos*. 2000.

3. Droogenbroek, M. and Benedett, R. *Techniques for Selective Encryption of Uncompressed and Compressed Images*. 2002.

4. Kailasanathan, C. and Naini, R. *Compression Performance of JPEG Encryption Scheme*. 2003.

5. Li, X., Knipe, J. and Cheng, H. *Image Compression and Encryption Using Tree Structures*. 1997.

6. Lookabaugh, T., Sicker, D., Keaton, D., Guoand, W. and Vedula, I. *Security Analysis of Selectively Encrypted MPEG-e Streams*. 2003.

7. Miano, J. *Compressed Image File Formats*. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

8. Norcen, R. and Uhl, A. *Selective Encryption of the JPEG2000 Bitstream*. 2003.

9. Pennebaker, W. and Mitchell J. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, New York, 1993.

10. Podesser, M., Schmidt, H. and Uhl, A. *Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments*. 2002.

11. Seo, Y., Kim, D., Yoo, J., Dey, S., Agrawal, A. *Wavelet Domain Image Encryption by Subband Selection and Data Bit Selection*. 2003.

Source Code

Team ISE

Image Selective Encryption

ISE Production Code Files

```
//--------------------------------------------------------------------------
//
//  ise.h
//
//  Authors: Joe Jarchow, Geoffrey Griffith, Joseph Kadhim, Shinya Daigaku
//           Andrew Pouzeshi
//
//  Sponsor: Tom Lookabaugh, Assistant Professor of Computer Science
//           University of Colorado
//
//  Senior Project: Team ISE (Image Selective Encryption)
//                  December 2003
//
//  For more information go to: http://128.138.75.184
//--------------------------------------------------------------------------
//
//  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.
//
//--------------------------------------------------------------------------
//
//  The purpose of this file is to define what functions and members
//  are to be exported for a programmer using the ISE class.  ISE
//  is a class defined to implement image selective encryption for
//  jpeg images.  class ise is intended to be the super class and
//  must be inherited by sub classes.  We have only implemented the
//  jpeg_ise class at this time but other classes could be implemented
//  following the outline used.  Along with constructors there are
//  are various functions for setting and getting the class members
//  each is defined in detail preceeding the appropriate function
//  in the ise.cpp file.
//
//--------------------------------------------------------------------------
#include <stdlib.h>
#include <iostream>
#include <fstream>
using std::ifstream;
using std::ofstream;

#ifndef ISE_H
#define ISE_H
    class ise
    {
        public:
                        ise(char*, char* = NULL, char* = NULL);
                        virtual ~ise();
                        virtual int encrypt_file() { return 0; }
                        virtual int decrypt_file() { return 0; }
                        int set_key(char*);
                        int set_input_file_name(char*);
                        int set_output_file_name(char*);
                        char* get_input_file_name();
                        char* get_output_file_name();
                protected:
                        ise();
                        int get_ise_file_type(char*);
                        int make_ise_file_name();
                        int make_output_file_name();
                        char* get_key();
            private:
                        char* input_file_name;
                        char* output_file_name;
                        char* key;
    };
#endif //ISE_H
```



```
#ifndef JPEG_ISE_H
#define JPEG_ISE_H
    class jpeg_ise : public ise
    {
        public:
                        jpeg_ise(char*, char* = NULL, char* = NULL);
                        ~jpeg_ise();
                        int encrypt_file();
                        int decrypt_file();
        protected:
                        jpeg_ise();
    };
#endif //JPEG_ISE_H
```

```
//--------------------------------------------------------------------------
//
//  ise.cpp
//
//  Authors: Joe Jarchow, Geoffrey Griffith, Joseph Kadhim, Shinya Daigaku
//           Andrew Pouzeshi
//
//  Sponsor: Tom Lookabaugh, Assistant Professor of Computer Science
//           University of Colorado
//
//  Senior Project: Team ISE (Image Selective Encryption)
//                  December 2003
//
//  For more information go to: http://128.138.75.184
//--------------------------------------------------------------------------
//
//  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.
//
//--------------------------------------------------------------------------
//
//  The purpose of this file is to define what functions and members
//  are to be exported for a programmer using the ISE class.  ISE
//  is a class defined to implement image selective encryption for
//  jpeg images.  class ise is intended to be the super class and
//  must be inherited by sub classes.  We have only implemented the
//  jpeg_ise class at this time but other classes could be implemented
//  following the outline used.  Along with constructors there are
//  are various functions for setting and getting the class members
//  each is defined in detail preceeding the appropriate function
//  in the ise.cpp file.
//
//--------------------------------------------------------------------------

#include <stdlib.h>
#include <string>
#include <iostream>
#include <stack>
#include <cstdlib>
#include "rijndael-api-fst.h" // use for block cipher encryption/decryption

using namespace std;
using std::cerr;
using std::endl;
using std::nothrow;

const int  JPEG_TYPE      = 1;     // specify jpeg ise
const char JPEG_FILE_TYPE = 'I';   // specify jpeg file type
const unsigned int  MIN_KEY_LENGTH = 32;    // minimum length of the key
const int  BUFFER_LENGTH  = 16;    // size of Rijndael encryption block

typedef unsigned char byte;

#include "ise.h"

//--------------------------------------------------------------------------
//
// Default Constructor
// Pre-conditions:      None.
// Post-conditions: None.
// Parameters:          None.
// Return values:       Constructor, no return type.
// Description:     Default constructor is not used by users.
//
//--------------------------------------------------------------------------
```



```
ise::ise()
{
}

//--------------------------------------------------------------------------
//
// Overloaded Constructor
// Pre-conditions:      The key must be a pointer to a character string.
// Post-conditions: An ISE object is created containing the specified
//                  data members.
// Parameters:      The first argument is a pointer to the key.
//                  The second argument is the name and path of the input file
//                  to be encrypted or decrypted.  The third argument is
//                  the file name and path for the output file generated by
//                  encryption or decryption.
// Return values:       Constructor, no return type.
// Description:     An ISE object is constructed with the data necessary to
//                  encrypt or decrypt a file.  This overloaded
//                  constructor only requires that the first argument
//                  be provided.  The second and third arguments are optional
//                  and will be set to a default value of NULL.
//
//--------------------------------------------------------------------------
ise::ise(char* key, char* input_file_name, char* output_file_name)
{
        size_t length;
        char * key_copy;
        char * temp;

        // check that the key in not NULL
        if (key == NULL)
        {
                exit(1);
        }

        // check that the input and output files are of type jpeg or ise
        char * index;
        if (input_file_name != NULL)
        {
                index = strstr(input_file_name, ".jp");
                if (index == NULL)
                {
                        index = strstr(input_file_name, ".JP");
                        if (index == NULL)
                        {
                                index = strstr(input_file_name, ".ise");
                                if (index == NULL)
                                {
                                        index = strstr(input_file_name, ".ISE");
                                        if (index == NULL)
                                        {
                                                exit(1);
                                        }
                                }
                        }
                }
        }

        if (output_file_name != NULL)
        {
                index = strstr(output_file_name, ".jp");
                if (index == NULL)
                {
                        index = strstr(output_file_name, ".JP");
                        if (index == NULL)
                        {
                                index = strstr(output_file_name, ".ise");
                                if (index == NULL)
                                {
```

```
                                        index = strstr(output_file_name, ".ISE");
                                        if (index == NULL)
                                        {
                                                exit(1);
                                        }
                                }
                        }
                }
        }

        // set the key
        length = strlen(key);
        key_copy = new (nothrow) char [length + 1];
        if (key_copy == NULL)
        {
                exit(1);
        }
        temp = new (nothrow) char [length * 2 + 1];
        if (temp == NULL)
        {
                exit(1);
        }
        strcpy(key_copy, key);

        // split each character into four bit values
        for (size_t i = 0; i < length; i++)
        {
            temp[i * 2] = key_copy[i] >> 4;
            key_copy[i] = key_copy[i] << 4;
            temp[i * 2 + 1] = key_copy[i] >> 4;
        }

        // convert four bit values to hexadecimal characters
        length = length * 2;
        temp[length] = '\0';
        for (size_t i = 0; i < length; i++)
        {
                switch((int)temp[i])
                {
                case 0:
                        temp[i] = '0';
                        break;
                case 1:
                        temp[i] = '1';
                        break;
                case 2:
                        temp[i] = '2';
                        break;
                case 3:
                        temp[i] = '3';
                        break;
                case 4:
                        temp[i] = '4';
                        break;
                case 5:
                        temp[i] = '5';
                        break;
                case 6:
                        temp[i] = '6';
                        break;
                case 7:
                        temp[i] = '7';
                        break;
                case -8:
                        temp[i] = '8';
                        break;
                case -7:
                        temp[i] = '9';
                        break;
```



```
                case -6:
                        temp[i] = 'a';
                        break;
                case -5:
                        temp[i] = 'b';
                        break;
                case -4:
                        temp[i] = 'c';
                        break;
                case -3:
                        temp[i] = 'd';
                        break;
                case -2:
                        temp[i] = 'e';
                        break;
                case -1:
                        temp[i] = 'f';
                        break;
                }
        }

        // extend the key length to 32 bytes
        if (length < MIN_KEY_LENGTH)
        {
                this->key = new (nothrow) char[MIN_KEY_LENGTH + 1];
                if (this->key == NULL)
                {
                        exit(1);
                }
                strcpy(this->key, temp);
                for (size_t i = length; i < MIN_KEY_LENGTH; i++)
                {
                        this->key[i] = '0';
                }
                this->key[MIN_KEY_LENGTH] = '\0';
        }
        else
        {
                this->key = new (nothrow) char[length + 1];
                if (this->key == NULL)
                {
                        exit(1);
                }
                strcpy(this->key, temp);
        }
        delete [] key_copy;
        delete [] temp;

        // set the input file name
        if (input_file_name != NULL)
        {
                length = strlen(input_file_name);
                this->input_file_name = new (nothrow) char[length + 1];
                if (this->input_file_name == NULL)
                {
                        exit(1);
                }
                strcpy(this->input_file_name, input_file_name);
        }
        else
        {
                this->input_file_name = NULL;
        }

        // set the output file name
        if (output_file_name != NULL)
        {
                length = strlen(output_file_name);
                this->output_file_name = new (nothrow) char[length + 1];
```

```
                if (this->output_file_name == NULL)
                {
                        exit(1);
                }
                strcpy(this->output_file_name, output_file_name);
        }
        else
        {
                this->output_file_name = NULL;
        }
}

ise::~ise()
{
        if (key != NULL)
        {
                delete [] key;
        }
        if (input_file_name != NULL)
        {
                delete [] input_file_name;
        }
        if (output_file_name != NULL)
        {
                delete [] output_file_name;
        }
}

//---------------------------------------------------------------------------
//
// Pre-conditions:  The key must point to a character string.
// Post-conditions:     The key will be set using the new string specified.
//                  Any previous information in key will be lost.
// Parameters:      The only argument to this method is a pointer to
//                  a character string containing the key information
//                  for either encryption or decryption.
// Return values:       An integer is returned indicating a success or failure.
//                  A zero will indicate a success.
//                  A one will indicate an invalid key.
//                                  A two will indicate a memory allocation
error.
// Description:     The method will use the specified character string to
//                  create a valid key to be used by the encryption or
//                  decryption methods.
//
//---------------------------------------------------------------------------
int ise::set_key(char* name)
{
        size_t length;
        char * name_copy;
        char * temp;

        // check that the key is not NULL
        if (name == NULL)
        {
                return 1;
        }

        length = strlen(name);
        name_copy = new (nothrow) char[length + 1];
        if (name_copy == NULL)
        {
                return 2;
        }
        temp = new (nothrow) char[length * 2 + 1];
        if (temp == NULL)
        {
                return 2;
        }
```



```
        strcpy(name_copy, name);

        // split each character into four bit values.
        for (size_t i = 0; i < length; i++)
        {
            temp[i * 2] = name_copy[i] >> 4;
            name_copy[i] = name_copy[i] << 4;
            temp[i * 2 + 1] = name_copy[i] >> 4;
        }

        length = length * 2;
        temp[length] = '\0';

        // convert four bit values to hexadecimal characters
        for (size_t i = 0; i < length; i++)
          {
            switch((int)temp[i])
              {
              case 0:
                temp[i] = '0';
                break;
              case 1:
                temp[i] = '1';
                break;
              case 2:
                temp[i] = '2';
                break;
              case 3:
                temp[i] = '3';
                break;
              case 4:
                temp[i] = '4';
                break;
              case 5:
                temp[i] = '5';
                break;
              case 6:
                temp[i] = '6';
                break;
              case 7:
                temp[i] = '7';
                break;
              case -8:
                temp[i] = '8';
                break;
              case -7:
                temp[i] = '9';
                break;
              case -6:
                temp[i] = 'a';
                break;
              case -5:
                temp[i] = 'b';
                break;
              case -4:
                temp[i] = 'c';
                break;
              case -3:
                temp[i] = 'd';
                break;
              case -2:
                temp[i] = 'e';
                break;
              case -1:
                temp[i] = 'f';
                break;
              }
          }
```

```
    // delete the previous key information
        delete [] key;

        // extend the key length to 32 bytes
        if (length < MIN_KEY_LENGTH)
        {
                key = new (nothrow) char[MIN_KEY_LENGTH + 1];
                if (key == NULL)
                {
                        return 2;
                }
                strcpy(key, temp);
                for (size_t i = length; i < MIN_KEY_LENGTH; i++)
                {
                        key[i] = '0';
                }
                key[MIN_KEY_LENGTH] = '\0';
        }
        else
        {
                key = new (nothrow) char[length + 1];
                if (key == NULL)
                {
                        return 2;
                }
                strcpy(key, temp);
        }

        delete [] name_copy;
        delete [] temp;

        return 0;
}

//-----------------------------------------------------------------------------
//
// Pre-conditions:      The name must be a pointer to a valid jpeg or ise file
//                      type.
// Post-conditions:     The input_file_name will be set using the new string
//                      specified.  Any previous data in input_file_name will
//                      be lost.
// Parameters:          The only argument to this method is a pointer to a
//                      character string containing the input_file_name,
//                      specifying the input file to encryption or decryption.
// Return values:       An integer is returned indicating a success or failure.
//                      A zero will indicate a success.
//                      A one will indicate an invalid input file name.
//                                      A two will indicate a memory allocation
error.
// Description:         This method is used to set the input_file_name.
//                      The method must be called prior to the encryption
//                      or decryption methods if they were not specified
//                      in the constructor.
//
//-----------------------------------------------------------------------------
int ise::set_input_file_name(char* name)
{
        size_t length;

        // check that the name is not NULL
        if (name == NULL)
        {
                return 1;
        }

        // check that the name is a jpeg or ise file type
        char * index;
        index = strstr(name, ".jp");
        if (index == NULL)
```



```
        {
                index = strstr(name, ".JP");
                if (index == NULL)
                {
                        index = strstr(name, ".ise");
                        if (index == NULL)
                        {
                                index = strstr(name, ".ISE");
                                if (index == NULL)
                                {
                                        return 1;
                                }
                        }
                }
        }

        // delete any previous input file information
        if (input_file_name != NULL)
        {
                delete [] input_file_name;
        }

        // set the input file name
        length = strlen(name);
        input_file_name = new (nothrow) char[length + 1];
        if (input_file_name == NULL)
        {
                return 2;
        }
        strcpy(input_file_name, name);

        return 0;
}

//-----------------------------------------------------------------------------
//
// Pre-conditions:      The name must be a pointer to a valid jpeg or ise file
//                      type.
// Post-conditions:     The output_file_name will be set using the new string
//                      specified.  Any previous data in output_file_name will
//                      be lost.
// Parameters:          The only argument to this method is a pointer to a
//                      character string containing the output_file_name,
//                      specifying the output file to encryption or decryption.
// Return values:       An integer is returned indicating a success or failure.
//                      A zero will indicate a success.
//                      A one will indicate an invalid output file name.
//                                      A two will indicate a memory allocation
error.
// Description:         This method is used to set the output_file_name.
//
//-----------------------------------------------------------------------------
int ise::set_output_file_name(char* name)
{
        size_t length;

        // check that the name is not NULL
        if (name == NULL)
        {
                return 1;
        }

        // check that the name is a jpeg or ise file type
        char * index;
        index = strstr(name, ".jp");
        if (index == NULL)
        {
                index = strstr(name, ".JP");
                if (index == NULL)
```

```
                {
                        index = strstr(name, ".ise");
                        if (index == NULL)
                        {
                                index = strstr(name, ".ISE");
                                if (index == NULL)
                                {
                                        return 1;
                                }
                        }
                }
        }

        // delete any previous output file information
        if (output_file_name != NULL)
        {
                delete [] output_file_name;
        }

        // set the output file name
        length = strlen(name);
        output_file_name = new (nothrow) char[length + 1];
        if (output_file_name == NULL)
        {
                return 2;
        }
        strcpy(output_file_name, name);

        return 0;
}

//-------------------------------------------------------------------------
//
// Pre-conditions:      None.
// Post-conditions:     None.
// Parameters:      None.
// Return values:   The method will return the input_file_name character string.
//                  If the input_file_name is not set, the method will return
//                  NULL.
// Description:     This is the accessor method for the input file name.
//
//-------------------------------------------------------------------------
char* ise::get_input_file_name()
{
        // check that the input file is not NULL
        if (input_file_name == NULL)
        {
                return NULL;
        }
        return input_file_name;
}

//-------------------------------------------------------------------------
//
// Pre-conditions:      None.
// Post-conditions:     None.
// Parameters:      None.
// Return values:   The method will return the output_file_name character string
.
//                  If the output_file_name is not set, the method will return
//                  NULL.
// Description:     This is the accessor method for the output file name.
//
//-------------------------------------------------------------------------
char* ise::get_output_file_name()
{
        // check that the output file is not NULL
        if (output_file_name == NULL)
        {
```



```
                return NULL;
        }
        return output_file_name;
}

//-------------------------------------------------------------------------
//
// Pre-conditions:      The name must be a pointer to a valid ISE file.
// Post-conditions:     None
// Parameters:      The only argument for this method is a pointer
//                  to a character string indicating the name of a
//                  valid ISE file.
// Return values:   The function will return an integer indicating
//                  the type of the original file from which the specified
//                  ISE file was created.
//                  0 will indicate an unknown or unimplemented file type.
//                  1 will indicate a jpeg file.
//                  2 will indicate a mp3 file.
//                  3 will indicate a zip file.
//                  The return values may be extended to accommodate other file
types.
// Description:     This method will return an integer corresponding to
//                  the original file type of an encrypted ISE file.
//
//-------------------------------------------------------------------------
int ise::get_ise_file_type(char* name)
{
        char the_type;

        ifstream ise_infs(name, ios::binary);

        // check that the file can be opened
        if (ise_infs.good() == false)
        {
                return 0;
        }
        // read the first byte from the ise file
        ise_infs.read(&the_type, sizeof(the_type));

        // check if the file is a jpeg ise
        if (the_type == '1')
        {
                return 1;
        }
        // check if the file is a mp3 ise
        if (the_type == '2')
        {
                return 2;
        }
        // check if the file is a zip ise
        if (the_type == '3')
        {
                return 3;
        }

        ise_infs.close();

        // otherwise the file is unknown
        return 0;
}

//-------------------------------------------------------------------------
//
// Pre-conditions:      The user of the class has previously set the input_file_
name.
// Post-conditions: The output_file_name data member points to a string with
//                  a file name and file path, based upon the string pointed
//                  to by the input_file_name.
// Parameters:      None.
```

```
// Return values:   An integer is returned indicating a success or failure.
//                  A zero will indicate a success.
//                  A one will indicate a failure.
// Description:     The file name and path created will be the same as the
//                  string pointed to by the input_file_name data member,
//                  except that the extension of the file will be changed
//                  to .ise.  If this file already exists, then a 0 will be
//                  added on to the end of the file name, just before the
//                  extension.  If this file already exists, we will keep
//                  incrementing this number and checking, until the new file
//                  name does not previously exist.
//
//-----------------------------------------------------------------------------
int ise::make_ise_file_name()
{
        char* index;
        // size equals length of extention number
        size_t length, size;
        // used to find the name extention number
        int number, temp, remainder, count, digit;
        char letter = '0';
        // stores name extention number
        stack<int> file_index;
        ifstream InFile;

        // set an ise file name from the input file name
        number = 0;
        length = strlen(input_file_name);
        output_file_name = new (nothrow) char[length + 1];
        if (output_file_name == NULL)
        {
                return 1;
        }
        strcpy(output_file_name, input_file_name);
        // check the jpeg file extention
        index = strstr(output_file_name, ".jp");
        // if file extention is ".JPG"
        if (index == NULL)
        {
                index = strstr(output_file_name, ".JP");
        }
        // check if not a jpeg file
        if (index == NULL)
        {
                return 1;
        }

        // add ise extention
        *(index+1) = 'i';
        *(index+2) = 's';
        *(index+3) = 'e';
        *(index+4) = '\0';

        InFile.open(output_file_name);

        // if file name already exists, make a new file name
        while (InFile.good())
        {
                InFile.close();
                number++;
                temp = number;
                // calculate name extention number
                while (temp != 0)
                {
                        remainder = temp % 10;
                        file_index.push(remainder);
                        temp = temp / 10;
                }
```



```
        // create output file name
        if (output_file_name != NULL)
        {
                delete [] output_file_name;
        }
        size = file_index.size();
        output_file_name = new (nothrow) char[length + size + 1];
        if (output_file_name == NULL)
        {
                return 1;
        }
        strcpy(output_file_name, input_file_name);
        index = strstr(output_file_name, ".jp");
        // if file extention is ".JPG"
        if (index == NULL)
        {
                index = strstr(output_file_name, ".JP");
        }
        count = 0;

        // convert top of stack to a character
        while (!file_index.empty())
        {
                digit = file_index.top();
                file_index.pop();
                switch (digit)
                {
                case 0:
                        letter = '0';
                        break;
                case 1:
                        letter = '1';
                        break;
                case 2:
                        letter = '2';
                        break;
                case 3:
                        letter = '3';
                        break;
                case 4:
                        letter = '4';
                        break;
                case 5:
                        letter = '5';
                        break;
                case 6:
                        letter = '6';
                        break;
                case 7:
                        letter = '7';
                        break;
                case 8:
                        letter = '8';
                        break;
                case 9:
                        letter = '9';
                        break;
                }
                // add extention number
                *(index + count) = letter;
                count++;
        }
        // add ise file extention
        *(index + size) = '.';
        *(index + size + 1) = 'i';
        *(index + size + 2) = 's';
        *(index + size + 3) = 'e';
        *(index + size + 4) = '\0';
```

```
                InFile.open(output_file_name);
        }
        return 0;
}

//---------------------------------------------------------------------------
//
// Pre-conditions:      The user of the class has previously set the input_file_
name.
// Post-conditions: The output_file_name data member points to a string with
//                  a file name and file path, based upon the string pointed
//                  to by the input_file_name.
// Parameters:      None.
// Return values:   An integer is returned indicating a success or failure.
//                  A zero will indicate a success.
//                  A one will indicate a failure.
// Description:     The file name and path created will be the same as the
//                  string pointed to by the input_file_name data member,
//                  except that the extension of the file will be changed
//                  to .jpg.  If this file already exists, then a 0 will be
//                  added on to the end of the file name, just before the
//                  extension.  If this file already exists, we will keep
//                  incrementing this number and checking, until the new file
//                  name does not previously exist.
//
//---------------------------------------------------------------------------
int ise::make_output_file_name()
{
        char* index;
        // size equals length of extention number
        size_t length, size;
        // used to find the name extention number
        int number, temp, remainder, count, digit;
        char letter = '0';
        // stores name extention number
        stack<int> file_index;
        ifstream InFile;

        // set an output file name from the ise file name
        number = 0;
        length = strlen(input_file_name);
        output_file_name = new (nothrow) char[length + 1];
        if (output_file_name == NULL)
        {
                return 1;
        }
        strcpy(output_file_name, input_file_name);
        // check the ise file extention
        index = strstr(output_file_name, ".is");
        // check if the extention is .ISE
        if (index == NULL)
        {
                index = strstr(input_file_name, ".IS");
        }
        // check if not a valid ise file
        if (index == NULL)
        {
                return 1;
        }
        // add jpeg extention
        *(index+1) = 'j';
        *(index+2) = 'p';
        *(index+3) = 'g';
        *(index+4) = '\0';

        InFile.open(output_file_name);

        // if file name already exists, make a new file name
        while (InFile.good())
```



```
        {
                InFile.close();
                number++;
                temp = number;
                // calculate name extention number
                while (temp != 0)
                {
                        remainder = temp % 10;
                        file_index.push(remainder);
                        temp = temp / 10;
                }

                // create output file name
                if (output_file_name != NULL)
                {
                        delete [] output_file_name;
                }
                size = file_index.size();
                output_file_name = new (nothrow) char[length + size + 1];
                if (output_file_name == NULL)
                {
                        return 1;
                }
                strcpy(output_file_name, input_file_name);
                index = strstr(output_file_name, ".is");
                // check if file extention is ".ISE"
                if (index == NULL)
                {
                        index = strstr(input_file_name, ".IS");
                }

                // index offset
                count = 0;

                // convert top of stack to a character
                while (!file_index.empty())
                {
                        digit = file_index.top();
                        file_index.pop();
                        switch (digit)
                        {
                        case 0:
                                letter = '0';
                                break;
                        case 1:
                                letter = '1';
                                break;
                        case 2:
                                letter = '2';
                                break;
                        case 3:
                                letter = '3';
                                break;
                        case 4:
                                letter = '4';
                                break;
                        case 5:
                                letter = '5';
                                break;
                        case 6:
                                letter = '6';
                                break;
                        case 7:
                                letter = '7';
                                break;
                        case 8:
                                letter = '8';
                                break;
                        case 9:
```

```
                                letter = '9';
                                break;
                        }
                        // add extention number
                        *(index + count) = letter;
                        count++;
                }
                // add jpeg extetion
                *(index + size) = '.';
                *(index + size + 1) = 'j';
                *(index + size + 2) = 'p';
                *(index + size + 3) = 'g';
                *(index + size + 4) = '\0';

                InFile.open(output_file_name);
        }
        return 0;
}

//---------------------------------------------------------------------------
//
// Pre-conditions:      None.
// Post-conditions:     None.
// Parameters:      None.
// Return values:   The method will return the key character string.
//                  If the key is not set, the method will return
//                  NULL.
// Description:     This is the accessor method for the key.
//
//---------------------------------------------------------------------------
char* ise::get_key()
{
        // check that the key is not NULL
        if (key == NULL)
        {
                return NULL;
        }
        return key;
}

//---------------------------------------------------------------------------
//
// Default Constructor
// Pre-conditions:      None.
// Post-conditions:     None.
// Parameters:          None.
// Return values:       Constructor, no return type.
// Description:     Default constructor is not used by users.
//
//---------------------------------------------------------------------------
jpeg_ise::jpeg_ise() : ise()
{
}

//---------------------------------------------------------------------------
//
// Overloaded Constructor
// Pre-conditions:      The key must be a pointer to a character string.
// Post-conditions: An JPEG_ISE object is created containing the specified
//                  data members.
// Parameters:      The first argument is a pointer to the key.
//                  The second argument is the name and path of the input file
//                  to be encrypted or decrypted.  The third argument is
//                  the file name and path for the output file generated by
//                  encryption or decryption.
// Return values:       Constructor, no return type.
// Description:     An ISE object is constructed with the data necessary to
//                  encrypt or decrypt a file.  This overloaded
//                  constructor only requires that the first argument
```



```
//                  be provided.  The second and third arguments are optional
//                  and will be set to a default value of NULL.
//
//---------------------------------------------------------------------------
jpeg_ise::jpeg_ise(char* key, char* input_file_name, char* output_file_name)
: ise(key, input_file_name, output_file_name)
{
}

jpeg_ise::~jpeg_ise()
{
}

//---------------------------------------------------------------------------
//
// Pre-conditions:      The input_file_name and key must be set using either
//                  the overloaded constructor or the
//                  set_input_file_name(char* name) and set_key(char* key)
//                  functions prior to calling this method.
//                  This code requires that the input and ouput file pointers
//                  are at the head of the file.
// Post-conditions: An encrypted file will be created with the name and path
//                  specified by the output_file_name data
//                  member.  If this data member is NULL, then a default file
//                  name will be created based upon the input_file_name
//                  data member.
// Parameters:      None.
// Return values:   An integer is returned indicating a success or failure.
//                  A zero will indicate a success.
//                  A one will indicate could not open input file name
//                  A two will indicate could not create ise file name
//                  A three will indicate could not open ise file
//                                      A four will indicate the jpeg file is no
t baseline
// Description:     The encrypt_file method will take a standard baseline
//                  compression JPEG file and selectively encrypt the
//                  Huffman Table frames found within the file.
//                  If the file already exists, the existing file will
//                  be overwritten.  A new, encrypted file will be
//                  created for the selectively encrypted JPEG image.
//
//---------------------------------------------------------------------------
int jpeg_ise::encrypt_file()
{
        // check if the input file exists
        ifstream infs(jpeg_ise::get_input_file_name(), ios::binary);
    if (infs.good() == false)
    {
                return 1;
    }

        // Check if ise_file_name is empty
        if (jpeg_ise::get_output_file_name() == NULL)
        {
                // create the ise output file
                jpeg_ise::make_ise_file_name();
                if (jpeg_ise::get_output_file_name() == NULL)
                {
                        return 2;
                }
        }

        // check if output file can open
    ofstream outfs(jpeg_ise::get_output_file_name(), ios::binary);
    if (outfs.good() == false)
    {
                return 3;
    }
```

```
    //output jpeg identifier to head of file
    char file_type;
    file_type = JPEG_FILE_TYPE;
    outfs.write(&file_type,sizeof(file_type));

    bool encrypt_huffman_table, encrypt_encoded_data;
    encrypt_huffman_table = encrypt_encoded_data = false;

    bool ff,inhuff,stop_encrypt, is_baseline, is_ffda;
    ff = inhuff = stop_encrypt = false;
    is_baseline = is_ffda = false;        //check if file contains FFC0, FFC4 or
FFDA
    int keyLength = 128;
    unsigned char plain_text[BUFFER_LENGTH];
    memset(plain_text,0,BUFFER_LENGTH);
    unsigned char cipher_text[BUFFER_LENGTH];
    memset(cipher_text,0,BUFFER_LENGTH);
    char cipher_text_output[BUFFER_LENGTH];
    memset(cipher_text_output,0,BUFFER_LENGTH);
    keyInstance keyinst;
    cipherInstance cipherinst;
        makeKey(&keyinst, DIR_ENCRYPT, keyLength, jpeg_ise::get_key());
    char iv[BUFFER_LENGTH];
    memset(iv,0,BUFFER_LENGTH);
    cipherInit(&cipherinst, MODE_ECB, iv);

    int pt_counter = 0;

    char b,c;
        // begin the ise selective encryption algorithm
    while (infs.read(&b,sizeof(b)))
        {
                // send unencrypted data to output file
        if (inhuff == false && stop_encrypt == false)
        {
            if ((byte)b == 0xFF)
            {
                outfs.write(&b,sizeof(b));
                infs.read(&b,sizeof(b));
                if ((byte)b == 0xC4 || (byte)b == 0xC0 )
                {
                                    // begin encrypting
                    inhuff = true;
                    is_baseline = true;
                }
                // non baseline jpeg marker
                else if ((byte)b == 0xC1 || (byte)b == 0xC2 || (byte)b == 0xC3 |
|                       (byte)b == 0xC5 || (byte)b == 0xC6 || (byte)b == 0xC7 |
|                       (byte)b == 0xC8 || (byte)b == 0xC9 || (byte)b == 0xCA |
|                       (byte)b == 0xCB || (byte)b == 0xCC || (byte)b == 0xCD |
|                       (byte)b == 0xCE || (byte)b == 0xCF)
                  {
                    return 4;
                  }
            }
            outfs.write(&b,sizeof(b));
        }

                // fill last buffer to be encrypted
        else if (inhuff == false && stop_encrypt == true)
        {
                    // fill last encryption buffer
            while (pt_counter < BUFFER_LENGTH)
            {
                plain_text[pt_counter++] = b;
```



```
                if(pt_counter < BUFFER_LENGTH) infs.read(&b,sizeof(b));
            }

                        // encrypt the buffer
            blockEncrypt(&cipherinst,&keyinst,plain_text,keyLength,cipher_text);
            // send encrypted data to output file
                        for (int i = 0; i < BUFFER_LENGTH; i++)
            {
                cipher_text_output[i]=(char)cipher_text[i];
                outfs.write(&cipher_text_output[i],sizeof(cipher_text_output[i])
);
            }
                        // reset the buffer
            memset(plain_text,0,BUFFER_LENGTH);
            memset(cipher_text,0,BUFFER_LENGTH);
            memset(cipher_text_output,0,BUFFER_LENGTH);
            pt_counter = 0;
                        // done encrypting
            stop_encrypt = false;
        }

                // encrypt huffman data of input file
        else
        {
                        // look for the begining of jpeg marker
            if ((byte)b == 0xFF)
            {
                infs.read(&c,sizeof(c));
                                // look for the non huffman marker
                if ((byte)c == 0xDA)
                {
                                        // go to fill last buffer
                    inhuff = false;
                    stop_encrypt = true;
                    is_ffda = true;
                }
                // check if file contains non baseline marker while encrypting
                if ((byte)c == 0xC1 || (byte)c == 0xC2 || (byte)c == 0xC3 ||
                    (byte)c == 0xC5 || (byte)c == 0xC6 || (byte)c == 0xC7 ||
                    (byte)c == 0xC8 || (byte)c == 0xC9 || (byte)c == 0xCA ||
                    (byte)c == 0xCB || (byte)c == 0xCC || (byte)c == 0xCD ||
                    (byte)c == 0xCE || (byte)c == 0xCF)
                  {
                    return 4;
                  }
                                // if huffman marker found, continue encryption
                if (pt_counter < BUFFER_LENGTH)
                {
                                        // add to the buffer
                    plain_text[pt_counter++] = b;
                }

                                // if huffman marker found and buffer is full, c
ontinue encryption
                else
                {
                                        // encrypt
                    blockEncrypt(&cipherinst,&keyinst,plain_text,keyLength,ciphe
r_text);
                    for (int i = 0; i < BUFFER_LENGTH; i++)
                    {
                                                // send to output file
                        cipher_text_output[i]=(char)cipher_text[i];
                        outfs.write(&cipher_text_output[i],sizeof(cipher_text_ou
tput[i]));
                    }
                                        // reset the buffer
                    memset(plain_text,0,BUFFER_LENGTH);
                    memset(cipher_text,0,BUFFER_LENGTH);
```

```
                    memset(cipher_text_output,0,BUFFER_LENGTH);
                    pt_counter = 0;
                    plain_text[pt_counter++] = b;
                }
                            // continue filling buffer
                if (pt_counter < BUFFER_LENGTH)
                {
                    plain_text[pt_counter++] = c;
                                    // encrypt if the buffer is full
                                    if(pt_counter == BUFFER_LENGTH)
                                    {
                                            blockEncrypt(&cipherinst,&keyins
t,plain_text,keyLength,cipher_text);
                                            for (int i = 0; i < BUFFER_LENGT
H; i++)
                                            {
                                                    // send to output file
                                                    cipher_text_output[i]=(c
har)cipher_text[i];
                                                    outfs.write(&cipher_text
_output[i],sizeof(cipher_text_output[i]));
                                            }
                                            // reset the buffer
                                            memset(plain_text,0,BUFFER_LENGT
H);
                                            memset(cipher_text,0,BUFFER_LENG
TH);
                                            memset(cipher_text_output,0,BUFF
ER_LENGTH);
                                            pt_counter = 0;
                                            stop_encrypt = false;
                                    }
                }

                            // if the buffer is full, encrypt and add c to b
uffer
                else
                {
                                    // encrypt
                    blockEncrypt(&cipherinst,&keyinst,plain_text,keyLength,ciphe
r_text);
                    for (int i = 0; i < BUFFER_LENGTH; i++)
                    {
                                            // send to output file
                        cipher_text_output[i]=(char)cipher_text[i];
                        outfs.write(&cipher_text_output[i],sizeof(cipher_text_ou
tput[i]));
                    }
                                    // reset the buffer
                    memset(plain_text,0,BUFFER_LENGTH);
                    memset(cipher_text,0,BUFFER_LENGTH);
                    memset(cipher_text_output,0,BUFFER_LENGTH);
                    pt_counter = 0;
                                    // add second half of marker to new buff
er
                    plain_text[pt_counter++] = c;
                }
            }

                // if no jpeg marker, fill a buffer and encrypt
            else
            {
                            // continue to fill buffer
                if (pt_counter < BUFFER_LENGTH)
                {
                    plain_text[pt_counter++] = b;
                }
                            // encrypt if the buffer is full
                else
```



```
                {
                    blockEncrypt(&cipherinst,&keyinst,plain_text,keyLength,ciphe
r_text);
                    for (int i = 0; i < BUFFER_LENGTH; i++)
                    {
                                            // send to output file
                        cipher_text_output[i]= (char)cipher_text[i];
                        outfs.write(&cipher_text_output[i],sizeof(cipher_text_ou
tput[i]));
                    }
                    // reset the buffer
                    memset(plain_text,0,BUFFER_LENGTH);
                    memset(cipher_text,0,BUFFER_LENGTH);
                    memset(cipher_text_output,0,BUFFER_LENGTH);
                    pt_counter = 0;
                                    // add b to new buffer
                    plain_text[pt_counter++] = b;
                }
            }
        }
        }

    if (!is_baseline || !is_ffda)
      {
        return 4;
      }

    infs.close();
    outfs.close();

    return 0;
}

//---------------------------------------------------------------------------
//
// Pre-conditions:      The input_file_name and key must be set using either
//                      the overloaded constructor or the
//                      set_input_file_name(char* name) and set_key(char* key)
//                      functions prior to calling this method.
//                      This code requires that the input and ouput file pointers
//                      are at the head of the file.
// Post-conditions: An decrypted file will be created with the name and path
//                      specified by the output_file_name data
//                      member.  If this data member is NULL, then a default file
//                      name will be created based upon the input_file_name
//                      data member.
// Parameters:          None.
// Return values:   An integer is returned indicating a success or failure.
//                      A zero will indicate a success.
//                      A one will indicate input file is not a jpeg ise file
//                      A two will indicate could not open ise file
//                      A three will indicate could not create output jpeg file
//                      A four will indicate could not open output jpeg file
// Description:         The decrypt_file method will take a JPEG ise file and
//                      selectively decrypt the Huffman Table frames found
//                      within the file.
//                      If the file already exists, the existing file will
//                      be overwritten.  A new, encrypted file will be
//                      created for the selectively new decrypted JPEG image.
//
//---------------------------------------------------------------------------
int jpeg_ise::decrypt_file()
{
        // check if input file is not a jpeg ise file
        if (jpeg_ise::get_ise_file_type(jpeg_ise::get_input_file_name()) != JPEG
_TYPE)
    {
                return 1;
    }
```

```
        ifstream infs(jpeg_ise::get_input_file_name(), ios::binary);

        // check if input file could not open
    if (infs.good() == false)
    {
                return 2;
    }

    // check if ise_file_name is NULL
    if (jpeg_ise::get_output_file_name() == NULL)
    {
                // create output jpeg file
                jpeg_ise::make_output_file_name();
                if (jpeg_ise::get_output_file_name() == NULL)
                {
                        return 3;
                }
    }

        // check if output file could not open
    ofstream outfs(jpeg_ise::get_output_file_name(), ios::binary);
    if (outfs.good() == false)
    {
                return 4;
    }

    //output jpeg identifier to head of file
    char file_type;
    infs.read(&file_type,sizeof(file_type));
        // check if file type of ise is
    /*if (file_type != '1')
    {
        return 1;
    }*/

    bool decrypt_huffman_table, decrypt_encoded_data;
    decrypt_huffman_table = decrypt_encoded_data = false;

    bool ff,inhuff,split_block;
    ff = inhuff = split_block = false;
    int keyLength = 128;
    unsigned char plain_text[BUFFER_LENGTH];
    memset(plain_text,0,BUFFER_LENGTH);
    unsigned char cipher_text[BUFFER_LENGTH];
    memset(cipher_text,0,BUFFER_LENGTH);
    char plain_text_output[BUFFER_LENGTH];
    memset(plain_text_output,0,BUFFER_LENGTH);
    keyInstance keyinst;
    cipherInstance cipherinst;
        makeKey(&keyinst, DIR_DECRYPT, keyLength, jpeg_ise::get_key());
    char iv[BUFFER_LENGTH];
    memset(iv,0,BUFFER_LENGTH);
    cipherInit(&cipherinst, MODE_ECB, iv);

    int ct_counter = 0;

    char b;
        // begin ise selective decryption algorithm
    while (infs.read(&b,sizeof(b)))
    {
                // send unencrypted data to output file
        if (inhuff == false && split_block == false)
        {
            if ((byte)b == 0xFF)
            {
                outfs.write(&b,sizeof(b));
                infs.read(&b,sizeof(b));
                if ((byte)b == 0xC4 || (byte)b == 0xC0)
                {
```



```
                    inhuff = true;
                }
            }
            outfs.write(&b,sizeof(b));
        }
                // if half of a jpeg marker was found
                // split block case
        else if (inhuff == true && split_block == true)
        {
                    // fill buffer to be decrypted
            while (ct_counter < BUFFER_LENGTH)
            {
                cipher_text[ct_counter++] = b;
                if(ct_counter < BUFFER_LENGTH) infs.read(&b,sizeof(b));
            }
                    // decrypt buffer
            blockDecrypt(&cipherinst,&keyinst,cipher_text,keyLength,plain_text);

                    // if first byte is not second half of huffman marker
                    if (plain_text[0] == 0xDA)
            {
                            // stop decryption
                inhuff = false;
            }
            split_block = false;

                    // send decrypted data to output file
                    for (int i = 0; i < BUFFER_LENGTH; i++)
            {
                plain_text_output[i]=(char)plain_text[i];
                outfs.write(&plain_text_output[i],sizeof(plain_text_output[i]));
            }
                    // reset the buffer
            memset(plain_text,0,BUFFER_LENGTH);
            memset(plain_text_output,0,BUFFER_LENGTH);
            memset(cipher_text,0,BUFFER_LENGTH);
            ct_counter = 0;
        }

                // in the huffman table
        else if(inhuff == true)
        {
                    // fill the buffer to be decrypted
            while (ct_counter < BUFFER_LENGTH)
            {
                cipher_text[ct_counter++] = b;
                            if(ct_counter < BUFFER_LENGTH) infs.read(&b,size
of(b));
            }

                    // decrypt the buffer
            blockDecrypt(&cipherinst,&keyinst,cipher_text,keyLength,plain_text);

                    // search through decrypted data
            for (int i = 0; i < BUFFER_LENGTH; i++)
            {
                            // if marker found
                if (plain_text[i] == 0xFF && i != 15)
                {
                                    // if not huffman marker
                    if (plain_text[i+1] == 0xDA)
                    {
                                            // stop decryption
                        inhuff = false;
                        break;
                    }
                }
                            // if half of jpeg marker found
                else if (plain_text[i] == 0xFF && i == 15)
```

```
                {
                                        // go to split block case
                    split_block = true;
                }
            }
                    // send decrypted data to output file
            for (int i = 0; i < BUFFER_LENGTH; i++)
            {
                plain_text_output[i]=(char)plain_text[i];
                outfs.write(&plain_text_output[i],sizeof(plain_text_output[i]));
            }
                    // reset the buffer
            memset(plain_text,0,BUFFER_LENGTH);
            memset(plain_text_output,0,BUFFER_LENGTH);
            memset(cipher_text,0,BUFFER_LENGTH);
            ct_counter = 0;
        }
    }

    infs.close();
    outfs.close();

    return 0;
}
```

ISE Manipulator Code Files

```
///-----------------------------------------------------------------------------
///
///  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
  }
}
```

ISE Website Code Files

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
<HTML>

<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <META http-equiv="Content-Language" content="en-US">
  <TITLE>Image Selective Encryption Home</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF" text="#000000" link="#0000FF"
vlink="#FF0000" alink="#FF0000" >

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P>
      <FONT size="+1" color="#FFFFFF">
      <B>University of Colorado Computer Science Department</B><br>
      2003-2004<br>
      </FONT>
      <FONT size="+2" color="#FFFFFF">
      <B>ISE JPEG Selective Encryption Home Page</B>
      </FONT>
    </P>
  </tr>
</TABLE>

<!-- These Script Commands import the menu bar.  The bar was created using
  Xsara Menu Maker.  This tool generated the javascript files used.
  These files were not written by the team.
-->

<P align="center">
  <SCRIPT src="images/menu/menu.js" type="text/javascript"></SCRIPT>
  <SCRIPT src="images/menu/isemenu.js" type="text/javascript"></SCRIPT>
</P>

<H1 align="center">
  <img src="images/ISE_icon_white.jpg" alt="Team ISE">
</H1>

<P align="center">
  <FONT color="#006600" face="Verdana" size="6">
  <B>Introduction</B>
  </FONT>
</P>

<P align="left">
  <FONT size="4">
  This website represents a team of students at the University of Colorado at
  Boulder Computer Science Department.  Under the sponsorship of Assistant
  Professor Tom Lookabaugh, Team ISE has researched and developed a method of
  Selective Encryption for the JPEG Baseline Compression Standard.  The
  research was done for the Senior Project required by the University of
  Colorado Department of Computer Science.
  <br>
  <br>
  This site serves as a distribution resource for the work done by Team ISE.
  This site contains a C++ class that implements the team's Selective
  Encryption method, a tool to manipulate portions of Baseline JPEG files,
  and all of the documentation developed by the team relevant to the
  product and their reaserch.
  <br>
  <br>
  </FONT>
</P>

<P align="center">
```



```
  <img src="images/ManipulatorShot1.jpg" alt="Manipulator">
</P>


<P align="center">
  <FONT color="#006600" face="Verdana" size="6">
  <B>Site Navigation</B>
  </FONT>
</P>

<P>
  <FONT size="4">
  This site is divided into nine sections:
  <br>
  <br>
  <a href="index.html" charset="ISO-8859-1"><B>Home</B></a>
  <br>
  The Home page, which is the current page, provides an introduction and
  relays information about the website.
  <br>
  <br>
  <a href="documents/ProjectProposal.pdf" target="_blank"
  charset="ISO-8859-1"><B>Project Proposal</B></a>
  <br>
  The project proposal is a PDF document giving a high level proposal for
  the Team ISE project.
  <br>
  <br>
  <a href="DocumentIndex.html" charset="ISO-8859-1"><B>Documentation</B></a>
  <br>
  This section contains all of the documentation produced for the project.
  <br>
  <br>
  <a href="Download.html" charset="ISO-8859-1"><B>Downloads</B></a>
  <br>
  This section contains links to download the C++ class as well as other
  tools developed by the team or relavant to the project.
  <br>
  <br>
  <a href="Sponsor.html" charset="ISO-8859-1"><B>Project Sponsor</B></a>
  <br>
  This section contains (minimal) information on the project sponsor.
  <br>
  <br>
  <a href="Team_ISE.html" charset="ISO-8859-1"><B>Team Info</B></a>
  <br>
  This section contains (minimal) information on the members of Team ISE.
  <br>
  <br>
  <a href="Links.html" charset="ISO-8859-1"><B>Links</B></a>
  <br>
  This section contains links to relevant websites.
  <br>
  <br>
  <a href="board/index.php" charset="ISO-8859-1"><B>ISE Message Board</B></a>
  <br>
  This section directs the user to a message board set up by the team.  The
  user can register and participate in any dialogs posted on the message
  board.  Team ISE would appreciate users posting any comments they may have
  on the reasearch done by the team.  The team would also appreciate the
  posting of any bugs found in the C++ library or the manipulator.
  <br>
  <br>
  <a href="Contact.html" charset="ISO-8859-1"><B>Contact Us</B></a>
  <br>
  This section provides contact information for Dr. Tom Lookabaugh and Team
  ISE.
  <br>
  <br>
```

```
  Users will find site a navigation menu under the header on every page,
  with the exception of the message board.  The user can navigate to the
  different sections of the website by clicking on these buttons.  Clicking
  on the top level button will direct the user to that page.  Some of the
  menu buttons, such as the Documentation Button and the Download Button,
  will display a submenu when higlited by the cursor.  This allows the user
  to jump directly to a specific document or download.  <B>Note:</B> When
  using the submenu to jump to documents a PDF file of the document will be
  opened.  If you want to download the document, click on the top-level
  Documentation button.  This will direct you to the document page where you
  can choose to either view or download the documents. Team ISE recommends
  new users visit the pages rather than using the quick links in the submenu.
  The pages contain  additional information which is useful to new users.
  </FONT>
</P>

<P>
  <FONT size="2">
  This project was done by
  <a href="http://www.colorado.edu" target="_blank">University of
  Colorado</a>
  students under the supervision of the
  <a href="http://www.cs.colorado.edu" target="_blank">Computer Science
  Department</a>.
  </FONT>
</P>

<P>
  <FONT size="2">
  This website is located on a sever at the University of Colorado at
  Boulder. Questions: Contact
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank">Tom Lookabaugh</a>
  or
  <a href="mailto:TeamISE@hotmail.com">TeamISE@hotmail.com</a>.
  </FONT>
</P>

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P align="center">
      <FONT color="#FFFFFF">
        <B>Team Image Selective Encryption Sponsored by Tom
        Lookabaugh</B><br>
        <B>Department of Computer Science</B><br>
        <B>University of Colorado at Boulder</B><br>
        <B>Boulder, CO 80309-0430</B><br>
        <B>HTML 4.01 Transitional</B><br>
        <B>Copyright &copy 2003-2004</B>
      </FONT>
    </P>
    <P align="right">
      <FONT color="#FFFFFF">
        Last Updated: <B>5/1/04</B>
      </FONT>
    </P>
  </tr>
</TABLE>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
<HTML>

<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <META http-equiv="Content-Language" content="en-US">
  <TITLE>Image Selective Encryption Documentation</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF" text="#000000" link="#0000FF"
vlink="#FF0000" alink="#FF0000">

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P>
      <FONT size="+1" color="#FFFFFF">
      <B>University of Colorado Computer Science Department</B><br>
      2003-2004<br>
      </FONT>
      <FONT size="+2" color="#FFFFFF">
      <B>ISE JPEG Selective Encryption Documentation Page</B>
      </FONT>
    </P>
  </tr>
</TABLE>

<!-- These Script Commands import the menu bar.  The bar was created using
  Xsara Menu Maker.  This tool generated the javascript files used.
  These files were not written by the team.
-->

<P align="center">
  <SCRIPT src="images/menu/menu.js" type="text/javascript"></SCRIPT>
  <SCRIPT src="images/menu/isemenu.js" type="text/javascript"></SCRIPT>
</P>

<H1 align="center">
  <img src="images/ISE_icon_white.jpg" alt="Team ISE">
</H1>

<P align="center">
  <FONT color="#006600" face="Verdana" size="6">
  <B>Documentation</B>
  </FONT>
</P>

<P>
  <FONT size="4">
  This page contains links to various documents produced by Team ISE while
  working on the project.  Access the different documents by clicking on the
  buttons below.  If using an Adobe PDF reader, the documents may be saved
  by clicking on the save button.  All documents are in PDF format.
  References may be made to all documentation, however ISE does not give
  anyone permission to change or modify these documents.
  </FONT>
</P>

<TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="20">
  <tr>
  <td>
    <a href="documents/Selective%20Encryption%20of%20JPEG%20images.doc"
    target="_blank">
      <img src="images/Selective_Encryption_of_JPEG.jpg" border="0"
      alt="Selective Encryption of JPEG Paper">
    </a>
  </td>
  <td>
```



```
    <P>
      <FONT size="4">
      This paper outlines team ISE's research and development of the ISE
      Selective Encryption for JPEG Baseline Compressed Images Algorithm.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="documents/ISEreference.zip">
      <img src="images/ISE_Reference_Document.jpg" border="0"
      alt="ISE Reference Manual">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      This is the reference manual for the ISE class which implements Team
      ISE's Baseline JPEG Selective Encryption algorithm.  Team ISE
      reccommends that you read this reference before using the class.
      Click on this button to download the reference document.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="documents/ISEmanpages.zip">
      <img src="images/ISE_man_pages.jpg" border="0" alt="ISE man pages">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      This is a link to the ISE class man pages.  These pages are helpful
      to users programming with the ISE class.  Clicking on the link will
      download the man pages.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="documents/ISE%20Manipulator%20Tutorial.pdf" target="_blank">
      <img src="images/Manipulator_Tutorial.jpg" border="0"
      alt="Manipulator Tutorial">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      This is a user tutorial for the JPEG Manipulator.  It is recommended
      that new users read this tutorial before using the JPEG Manipulator.
      <a href="documents/ManipulatorTutorial.zip">
        Download
      </a>
      the .ZIP file of the Manipulator Tutorial.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="documents/ISE%20Manipulator%20Manual.pdf" target="_blank">
      <img src="images/Manipulator_Reference.jpg" border="0"
      alt="Manipulator Reference">
    </a>
  </td>
```

```
<td>
  <P>
    <FONT size="4">
    This is the reference document for the JPEG Manipulator.  Any
    information needed to use the JPEG Manipulator and any of its
    components can be found in this document.
    <a href="documents/ManipulatorReference.zip">
      Download
    </a>
    the .ZIP file of the Manipulator Reference.
    </FONT>
  </P>
</td>
</tr>
<tr>
<td>
  <a href="documents/ISE_Developers_Reference/developer.htm"
  target="_blank">
    <img src="images/Developers_Reference.jpg" border="0"
    alt="Developer's Reference">
  </a>
</td>
<td>
  <P>
    <FONT size="4">
    This is the developer's reference document for the Team ISE Project.
    Any one interested in extending or modifying the code should read
    this document before doing so.
    <a href="documents/ISE_Developers_Reference.zip">
      Download
    </a>
    the .ZIP file of the Developer's Reference.
    </FONT>
  </P>
</td>
</tr>
<tr>
<td>
  <a href="documents/ISEFinalRequirements.pdf" target="_blank">
     <img src="images/Button_Requirements.jpg"  border="0"
     alt="Requirements Doc">
  </a>
</td>
<td>
  <P>
    <FONT size="4">
    This document outlines the requirements put foreward by Project
    Sponsor Tom Lookabaugh.  Team ISE followed these requirements as
    guidelines when developing the C++ class, the JPEG Manipulator
    and the website.
    <a href="documents/ISEFinalRequirements.zip">
      Download
    </a>
    the .ZIP file of the Requirements document.
    </FONT>
  </P>
</td>
</tr>
<tr>
<td>
  <a href="documents/ISEPrototypePlan.pdf" target="_blank">
    <img src="images/Button_Prototype_Plan.jpg" border="0"
    alt="Prototype Plan">
  </a>
</td>
<td>
  <P>
    <FONT size="4">
    This document outlines the plan devised by Team ISE to create a
```



```
    prototype for the project.
    <a href="documents/ISEPrototypePlan.zip">
      Download
    </a>
    the .ZIP file of the Prototype Plan.
    </FONT>
  </P>
</td>
</tr>
<tr>
<td>
  <a href="documents/ISESystemArchitectureDesign.pdf" target="_blank">
    <img src="images/Button_System_Architecture_Design.jpg"
    border="0" alt="Sys Arch Doc">
  </a>
</td>
<td>
  <P>
    <FONT size="4">
    This document give a detailed explanation of the System Architecture
    Design of the ISE project.  The document outlines the architecture
    of the C++ class, the JPEG Manipulator, and the website.  This
    document is helpful to anyone wishing to extend the C++ class to
    perform Selective Encryption methods on other types of file formats.
    <a href="documents/ISESystemArchitectureDesign.zip">
      Download
    </a>
    the .ZIP file of the System Architecuture document.
    </FONT>
  </P>
</td>
</tr>
<tr>
<td>
  <a href="documents/DesignSpecFinal.pdf" target="_blank">
    <img src="images/design_document.jpg" border="0" alt="Design Doc">
  </a>
</td>
<td>
  <P>
    <FONT size="4">
    This document contains very detailed information about the design of
    the C++ class, the JPEG Manipulator, and the website.  This document
    provides useful information on all of products being delivered by
    Team ISE.  Anyone interested in the design of the class or modifying
    the class in anyway should read this document.
    <a href="documents/ISEDesign.zip">
      Download
    </a>
    the .ZIP file of the Design document.
    </FONT>
  </P>
</td>
</tr>
<tr>
<td>
  <a href="documents/TestPlan.pdf" target="_blank">
    <img src="images/Test_plan_doc.jpg" border="0" alt="Test Plan Doc">
  </a>
</td>
<td>
  <P>
    <FONT size="4">
    This document contains detailed information about the test plan
    executed by Team ISE on all of its software products.  The document
    also reports the status of the performed tests.
    <a href="documents/ISETestPlan.zip">
      Download
    </a>
```

```
      the .ZIP file of the Test Plan document.
      </FONT>
    </P>
  </td>
  </tr>
</TABLE>

<P>
  <FONT size="2">
  This project was done by
  <a href="http://www.colorado.edu" target="_blank">University of
  Colorado</a>
  students under the supervision of the
  <a href="http://www.cs.colorado.edu" target="_blank">Computer Science
  Department</a>.
  </FONT>
</P>

<P>
  <FONT size="2">
  This website is located on a sever at the University of Colorado at
  Boulder. Questions: Contact
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank">Tom Lookabaugh</a>
  or
  <a href="mailto:TeamISE@hotmail.com">TeamISE@hotmail.com</a>.
  </FONT>
</P>

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P align="center">
      <FONT color="#FFFFFF">
        <B>Team Image Selective Encryption Sponsored by Tom
        Lookabaugh</B><br>
        <B>Department of Computer Science</B><br>
        <B>University of Colorado at Boulder</B><br>
        <B>Boulder, CO 80309-0430</B><br>
        <B>HTML 4.01 Transitional</B><br>
        <B>Copyright &copy 2003-2004</B>
      </FONT>
    </P>
    <P align="right">
      <FONT color="#FFFFFF">
        Last Updated: <B>5/1/04</B>
      </FONT>
    </P>
  </tr>
</TABLE>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
<HTML>

<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <META http-equiv="Content-Language" content="en-US">
  <TITLE>ISE Downloads</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF" text="#000000" link="#0000FF"
vlink="#FF0000" alink="#FF0000">

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P>
      <FONT size="+1" color="#FFFFFF">
      <B>University of Colorado Computer Science Department</B><br>
      2003-2004<br>
      </FONT>
      <FONT size="+2" color="#FFFFFF">
      <B>ISE JPEG Selective Encryption Download Page</B>
      </FONT>
    </P>
  </tr>
</TABLE>

<!-- These Script Commands import the menu bar.  The bar was created using
  Xsara Menu Maker.  This tool generated the javascript files used.
  These files were not written by the team.
-->

<P align="center">
  <SCRIPT src="images/menu/menu.js" type="text/javascript"></SCRIPT>
  <SCRIPT src="images/menu/isemenu.js" type="text/javascript"></SCRIPT>
</P>

<H1 align="center">
  <img src="images/ISE_icon_white.jpg" alt="Team ISE">
</H1>

<P align="center">
  <FONT color="#006600" face="Verdana" size="6">
  <B>ISE Downloads</B>
  </FONT>
</P>

<P>
  <FONT size="4">
  This page contains links to download ISE software.  The ISE class
  implementing Team ISE's selective encryption algorithm for JPEG
    compression and the ISE JPEG Manipulator can be downloaded from here.
    Also available are downloads for the Microsoft .NET framework (required
    to use the manipulator) and Alpha and Beta Test versions of our code.
  <br>
  <br>
  All documentation for the code can be viewed and downloaded from the
  <a href="DocumentIndex.html">ISE Documentation Page</a>.  It is
  recommended that you read the man pages and the user tutorials before
  using the code or software.  If you wish to modify the code in any way,
  you should also read the System Architecture document, the Design
  document, and the Developer's manual.
  <br>
  <br>
  It is the hope of Team ISE that the ISE class will be extended to perform
  selective encryption on many different compressed file formats.  Team
  ISE has laid the ground work for such a library and has taken the first
  step by devising and implementing a selective encryption technique for
```



```
  the JPEG Baseline Compression Standard.  The ISE class was developed so
  any other methods could easily inherit from the ISE class, thus
  developers would (in most cases) only have to devise and implement the
  selective encryption algorithms for different file formats.
  </FONT>
</P>

<P align="center">
  <FONT color="#006600" face="Verdana" size="6">
  <B>Licensing</B>
  </FONT>
</P>

<P>
  <FONT size="4">
  All of the code provided by Team ISE is open source.  Users are allowed
  to modify the code in any way.  Some users may want to use a different
  encryption algorithm than the AES algorithm included in the ISE class.
  Any changes made to the ISE code must adhere to the Team ISE licensing
  agreement.  Users may not remove the ISE name, the team member's names,
  nor license from any changed code.  However, users may add their own
  name to any changes made. Users must also follow the Licensing agreement
  in the Rijndael code, which was not written by Team ISE.
  <br>
  <br>
  All ISE code is provided for non-commercial use.  If you wish to use ISE
  code in commercial applications, you must first contact
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank">Dr. Tom Lookabaugh</a>
  and receive his permission and the permission of the
  <a href="http://www.colorado.edu/" target="_blank">University of Colorado
  </a></FONT>
</P>

<TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="20">
  <tr>
  <td>
    <a href="code/code.zip">
      <img src="images/Button_production_code.jpg"
      alt= "Rijndael Algoithm" border="0">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      Click on this link to downlode the ISE class code which
      implements the ISE selective encryption technique for the JPEG
      Baseline Compression.  This code is open source, and may be
      modified by the user.  Users are not licesned to use this code
      for commercial use without satisfying the conditions stated in
        the Licencing section of this page.  Team ISE recomends that you
        read the documentation before using the code.
      </FONT>
    </P>
  </td>
  </tr>
</TABLE>

<P align="center">
  <img src="images/ISE_Production_Code_ScreenShot.jpg"
  alt="ISE Production Code">
</P>

<TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="20">
  <tr>
  <td>
    <a href="code/ISEManipulator107.zip">
      <img src="images/Button_manipulator.jpg"
      alt= "ISE Manipulator" border="0">
```

```
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      Click on this link to download the ISE JPEG Manipulator. Please
      refer to the user tutorial for information on how to use the
      Manipulator.  The ISE JPEG Manipulator was developed as a
      research tool to aid Team ISE in creating a JPEG Baseline
      selective encryption method.  However, Team ISE is providing
      the JPEG Manipulator to interested users.
      <B>Note:</B> The JPEG Manipulator requires the Microsoft .NET
      Framework to run.  If you do not have the
      <a href="code/dotnetfx.exe">.NET Framework</a>,
      you can download it by clicking on the link below.
      </FONT>
    </P>
  </td>
  </tr>
</TABLE>

<P align="center">
  <img src="images/ManipulatorShot2.jpg" alt="ISE Manipulator">
</P>

<TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="20">
  <tr>
  <td>
    <a href="code/dotnetfx.exe">
      <img src="images/Button_dot_net.jpg"
      alt= ".Net Framework" border="0">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      Click on this link to download the Microsoft .Net Framework.
      The .NET framework is required to run the ISE JPEG Manipulator.
      You must follow Microsoft's licensing agreements on the .NET
      framework.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="code/iseAlpha.zip">
      <img src="images/alpha_test_code.jpg"
      alt= "Alpha Test Code" border="0">
    </a>
  </td>
    <td>
    <P>
      <FONT size="4">
      Click on this link to download Team ISE's Alpha Test version
      of the software. This code is buggy and differs from the
      completed ISE code available above.  However, Team ISE is
      providing it for those who are interested.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="code/iseBeta.zip">
      <img src="images/ISE_Beta_Release.jpg"
      alt= "Alpha Test Code" border="0">
    </a>
  </td>
```



```
    <td>
    <P>
      <FONT size="4">
      Click on this link to download Team ISE's Beta Test version of
      the software. This code is less buggy than the Alpha Test
      release, however it still differs from the completed ISE code
      available above.  Team ISE is providing it for those who are
      interested.
      </FONT>
    </P>
  </td>
  </tr>
</TABLE>

<P>
  <FONT size="2">
  This project was done by
  <a href="http://www.colorado.edu" target="_blank">University of Colorado
  </a>
  students under the supervision of the
  <a href="http://www.cs.colorado.edu" target="_blank">Computer Science
  Department</a>.
  </FONT>
</P>

<P>
  <FONT size="2">
  This website is located on a sever at the University of Colorado at
  Boulder. Questions: Contact
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank">Tom Lookabaugh</a>
  or
  <a href="mailto:TeamISE@hotmail.com">TeamISE@hotmail.com</a>.
  </FONT>
</P>

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P align="center">
      <FONT color="#FFFFFF">
        <B>Team Image Selective Encryption Sponsored by Tom
        Lookabaugh</B><br>
        <B>Department of Computer Science</B><br>
        <B>University of Colorado at Boulder</B><br>
        <B>Boulder, CO 80309-0430</B><br>
        <B>HTML 4.01 Transitional</B><br>
        <B>Copyright &copy 2003-2004</B>
      </FONT>
    </P>
    <P align="right">
      <FONT color="#FFFFFF">
        Last Updated: <B>5/1/04</B>
      </FONT>
    </P>
  </tr>
</TABLE>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
<HTML>

<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <META http-equiv="Content-Language" content="en-US">
  <TITLE>ISE Related Links</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF" text="#000000" link="#0000FF"
vlink="#FF0000" alink="#FF0000">

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P>
      <FONT size="+1" color="#FFFFFF">
      <B>University of Colorado Computer Science Department</B><br>
      2003-2004<br>
      </FONT>
      <FONT size="+2" color="#FFFFFF">
      <B>ISE JPEG Selective Encryption Links Page</B>
      </FONT>
    </P>
  </tr>
</TABLE>

<!-- These Script Commands import the menu bar.  The bar was created using
  Xsara Menu Maker.  This tool generated the javascript files used.
  These files were not written by the team.
-->

<P align="center">
  <SCRIPT src="images/menu/menu.js" type="text/javascript"></SCRIPT>
  <SCRIPT src="images/menu/isemenu.js" type="text/javascript"></SCRIPT>
</P>

<H1 align="center">
  <img src="images/ISE_icon_white.jpg" alt="Team ISE">
</H1>

<TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="20">
  <tr>
  <td>
    <a href="http://www.ijg.org" target="_blank">
      <img src="images/Button_ijg.jpg" alt= "ijg.org" border="0">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      Clicking on this button will open the Independent JPEG Group's (IJG)
      site in a new browser window.  IJG writes and distributes a free JPEG
      image compression library.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="http://www.esat.kuleuven.ac.be/~rijmen/rijndael/"
    target="_blank">
      <img src="images/Button_rij_link.jpg" alt= "Rijndael Algoithm"
      border="0">
    </a>
  </td>
  <td>
    <P>
```



```
      <FONT size="4">
      Clicking on this button will open a web page containing information on
      the Rijndael block cipher in a new browser window.  This block cipher
      is used in the C++ class for selectively encrypting JPEG Baseline
      Compression files. Implementations of the Rijndael AES block cipher
      can be downloaded from this page.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="http://itd.colorado.edu/lookabaugh/" target="_blank">
      <img src="images/Button_Project_Sponsor.jpg" alt= "Sponsor's Home"
      border="0">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      Clicking on this button will open project sponsor Dr. Tom Lookabaugh's
      site in a new browser window.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="http://www.colorado.edu/" target="_blank">
      <img src="images/University_of_Colorado.jpg"
      alt= "University of Colorado" border="0">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      Clicking on this button will open the University of Colorado home page
      in a new browser window.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="http://www.cs.colorado.edu/" target="_blank">
      <img src="images/CU_Department_of_Computer_Science.jpg"
      alt= "CU CS Dept." border="0">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      Clicking on this button will open the University of Colorado Computer
      Science Department's home page in a new browser window.
      </FONT>
    </P>
  </td>
  </tr>
  <tr>
  <td>
    <a href="http://itd.colorado.edu/" target="_blank">
      <img src="images/CU_Department_of_ITD.jpg" alt= "ITD" border="0">
    </a>
  </td>
  <td>
    <P>
      <FONT size="4">
      Clicking on this button will open the University of Colorado
      Interdisciplinary Telecommunications Department's home page in a new
```

```
      browser window.
      </FONT>
    </P>
  </td>
  </tr>
</TABLE>

<P>
  <FONT size="2">
  This project was done by
  <a href="http://www.colorado.edu" target="_blank">University of
  Colorado</a>
  students under the supervision of the
  <a href="http://www.cs.colorado.edu" target="_blank">Computer Science
  Department</a>.
  </FONT>
</P>

<P>
  <FONT size="2">
  This website is located on a sever at the University of Colorado at
  Boulder. Questions: Contact
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank">Tom Lookabaugh</a>
  or
  <a href="mailto:TeamISE@hotmail.com">TeamISE@hotmail.com</a>.
  </FONT>
</P>

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P align="center">
      <FONT color="#FFFFFF">
        <B>Team Image Selective Encryption Sponsored by Tom
        Lookabaugh</B><br>
        <B>Department of Computer Science</B><br>
        <B>University of Colorado at Boulder</B><br>
        <B>Boulder, CO 80309-0430</B><br>
        <B>HTML 4.01 Transitional</B><br>
        <B>Copyright &copy 2003-2004</B>
      </FONT>
    </P>
    <P align="right">
      <FONT color="#FFFFFF">
        Last Updated: <B>5/1/04</B>
      </FONT>
    </P>
  </tr>
</TABLE>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
<HTML>

<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <META http-equiv="Content-Language" content="en-US">
  <TITLE>Contact Information</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF" text="#000000" link="#0000FF"
vlink="#FF0000" alink="#FF0000">

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P>
      <FONT size="+1" color="#FFFFFF">
      <B>University of Colorado Computer Science Department</B><br>
      2003-2004<br>
      </FONT>
      <FONT size="+2" color="#FFFFFF">
      <B>ISE JPEG Selective Encryption Contact Page</B>
      </FONT>
    </P>
  </tr>
</TABLE>

<!-- These Script Commands import the menu bar.  The bar was created using
  Xsara Menu Maker.  This tool generated the javascript files used.
  These files were not written by the team.
-->

<P align="center">
  <SCRIPT src="images/menu/menu.js" type="text/javascript"></SCRIPT>
  <SCRIPT src="images/menu/isemenu.js" type="text/javascript"></SCRIPT>
</P>

<H1 align="center">
  <img src="images/ISE_icon_white.jpg" alt="Team ISE">
</H1>

<P align="center">
  <FONT color="#006600" face="Verdana" size="6">
  <B>Contact Information</B>
  </FONT>
</P>

<P>
  <FONT size="4">
  As this is a senior project and many members of the project will be
  graduating in May 2004, contact should be made with
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank"><B>Dr. Tom Lookabaugh</B></a>.
  <br>
  <br>
  However, you can also contact Team ISE members at
  <a href="mailto:TeamISE@hotmail.com"><B>TeamISE@hotmail.com</B></a>.
  <br>
  <br>
  <B>Note:</B> A quicker response will most likely be made if you contact
  Dr. Tom Lookabaugh rather than Team ISE.  However, Team ISE would be
  happy to address any comments or concerns you may have.
  </FONT>
</P>

<P>
  <FONT size="2">
  This project was done by
```



```
  <a href="http://www.colorado.edu" target="_blank">University of
  Colorado</a>
  students under the supervision of the
  <a href="http://www.cs.colorado.edu" target="_blank">Computer Science
  Department</a>.
  </FONT>
</P>

<P>
  <FONT size="2">
  This website is located on a sever at the University of Colorado at
  Boulder. Questions: Contact
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank">Tom Lookabaugh</a>
  or
  <a href="mailto:TeamISE@hotmail.com">TeamISE@hotmail.com</a>.
  </FONT>
</P>

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P align="center">
      <FONT color="#FFFFFF">
        <B>Team Image Selective Encryption Sponsored by Tom
        Lookabaugh</B><br>
        <B>Department of Computer Science</B><br>
        <B>University of Colorado at Boulder</B><br>
        <B>Boulder, CO 80309-0430</B><br>
        <B>HTML 4.01 Transitional</B><br>
        <B>Copyright &copy 2003-2004</B>
      </FONT>
    </P>
    <P align="right">
      <FONT color="#FFFFFF">
        Last Updated: <B>5/1/04</B>
      </FONT>
    </P>
  </tr>
</TABLE>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
<HTML>

<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <META http-equiv="Content-Language" content="en-US">
  <TITLE>Image Selective Encryption Sponsor</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF" text="#000000" link="#0000FF"
vlink="#FF0000" alink="#FF0000">

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P>
      <FONT size="+1" color="#FFFFFF">
      <B>University of Colorado Computer Science Department</B><br>
      2003-2004<br>
      </FONT>
      <FONT size="+2" color="#FFFFFF">
      <B>ISE JPEG Selective Encryption Sponsor</B>
      </FONT>
    </P>
  </tr>
</TABLE>

<!-- These Script Commands import the menu bar.  The bar was created using
  Xsara Menu Maker.  This tool generated the javascript files used.
  These files were not written by the team.
-->

<P align="center">
  <SCRIPT src="images/menu/menu.js" type="text/javascript"></SCRIPT>
  <SCRIPT src="images/menu/isemenu.js" type="text/javascript"></SCRIPT>
</P>

<H1 align="center">
  <img src="images/ISE_icon_white.jpg" alt="Team ISE">
</H1>

<P align="center">
  <FONT color="#006600" face="Verdana" size="6">
  <B>Sponsor Information</B>
  </FONT>
</P>

<P>
  <FONT size="4">
  Team ISE's Selective Encrption Project is being sponsored by Associate
  Professor <B>Tom Lookabaugh</B>.  In addition to sponsoring selective
  encryption research on the JPEG Baseline Compression format, Professor
  Lookabaugh has also done selective encryption reserach on the MPEG-2
  stream and speech coding.  Professor Lookabaugh is the Faculty Director
  for the
  <a href="http://itd.colorado.edu" target="_blank">Interdisciplinary
  Telecommunications Program</a>
  at the University of Colorado at Boulder. To visit Professor Lookabaugh's
  website click on
  the Project Sponsor button below.
  </FONT>
</P>

<P align="center">
  <img src="images/tom_lookabaugh[1].png" alt= "Tom Lookabaugh"
  align="middle">
  <br>
    <a href="http://itd.colorado.edu/lookabaugh/" target="_blank">
```



```
    <img src="images/Button_Project_Sponsor.jpg" alt= "Sponsor's Home"
    border="0" align="middle">
  </a>
</P>

<P>
  <FONT size="2">
  This project was done by
  <a href="http://www.colorado.edu" target="_blank">University of
  Colorado</a>
  students under the supervision of the
  <a href="http://www.cs.colorado.edu" target="_blank">Computer Science
  Department</a>.
  </FONT>
</P>

<P>
  <FONT size="2">
  This website is located on a sever at the University of Colorado at
  Boulder. Questions: Contact
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank">Tom Lookabaugh</a>
  or
  <a href="mailto:TeamISE@hotmail.com">TeamISE@hotmail.com</a>.
  </FONT>
</P>

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P align="center">
      <FONT color="#FFFFFF">
        <B>Team Image Selective Encryption Sponsored by Tom
        Lookabaugh</B><br>
        <B>Department of Computer Science</B><br>
        <B>University of Colorado at Boulder</B><br>
        <B>Boulder, CO 80309-0430</B><br>
        <B>HTML 4.01 Transitional</B><br>
        <B>Copyright &copy 2003-2004</B>
      </FONT>
    </P>
    <P align="right">
      <FONT color="#FFFFFF">
        Last Updated: <B>5/1/04</B>
      </FONT>
    </P>
  </tr>
</TABLE>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
<HTML>

<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <META http-equiv="Content-Language" content="en-US">
  <TITLE>Team Image Selective Encryption</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF" text="#000000" link="#0000FF"
vlink="#FF0000" alink="#FF0000">

<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P>
      <FONT size="+1" color="#FFFFFF">
      <B>University of Colorado Computer Science Department</B><br>
      2003-2004<br>
      </FONT>
      <FONT size="+2" color="#FFFFFF">
      <B>ISE JPEG Selective Encryption Team Page</B>
      </FONT>
    </P>
  </tr>
</TABLE>

<!-- These Script Commands import the menu bar.  The bar was created using
  Xsara Menu Maker.  This tool generated the javascript files used.
  These files were not written by the team.
-->

<P align="center">
  <SCRIPT src="images/menu/menu.js" type="text/javascript"></SCRIPT>
  <SCRIPT src="images/menu/isemenu.js" type="text/javascript"></SCRIPT>
</P>

<H1 align="center">
  <img src="images/ISE_icon_white.jpg" alt="Team ISE">
</H1>

<P align="center">
  <img src="images/team%20ISE/TeamISE.jpg" height="384" Width="512"
  alt="Team ISE">
</P>

<P>
  <FONT size="2">
  This project was done by
  <a href="http://www.colorado.edu" target="_blank">University of
  Colorado</a>
  students under the supervision of the
  <a href="http://www.cs.colorado.edu" target="_blank">Computer Science
  Department</a>.
  </FONT>
</P>

<P>
  <FONT size="2">
  This website is located on a sever at the University of Colorado at
  Boulder. Questions: Contact
  <a href="http://www.cs.colorado.edu/people/tom_lookabaugh.html"
  target="_blank">Tom Lookabaugh</a>
  or
  <a href="mailto:TeamISE@hotmail.com">TeamISE@hotmail.com</a>.
  </FONT>
</P>
```



```
<TABLE bgcolor="#003300" align="center" width="100%">
  <tr>
  <td>
    <P align="center">
      <FONT color="#FFFFFF">
        <B>Team Image Selective Encryption Sponsored by Tom
        Lookabaugh</B><br>
        <B>Department of Computer Science</B><br>
        <B>University of Colorado at Boulder</B><br>
        <B>Boulder, CO 80309-0430</B><br>
        <B>HTML 4.01 Transitional</B><br>
        <B>Copyright &copy 2003-2004</B>
      </FONT>
    </P>
    <P align="right">
      <FONT color="#FFFFFF">
        Last Updated: <B>5/1/04</B>
      </FONT>
    </P>
  </tr>
</TABLE>

</BODY>
</HTML>
```