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

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

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.

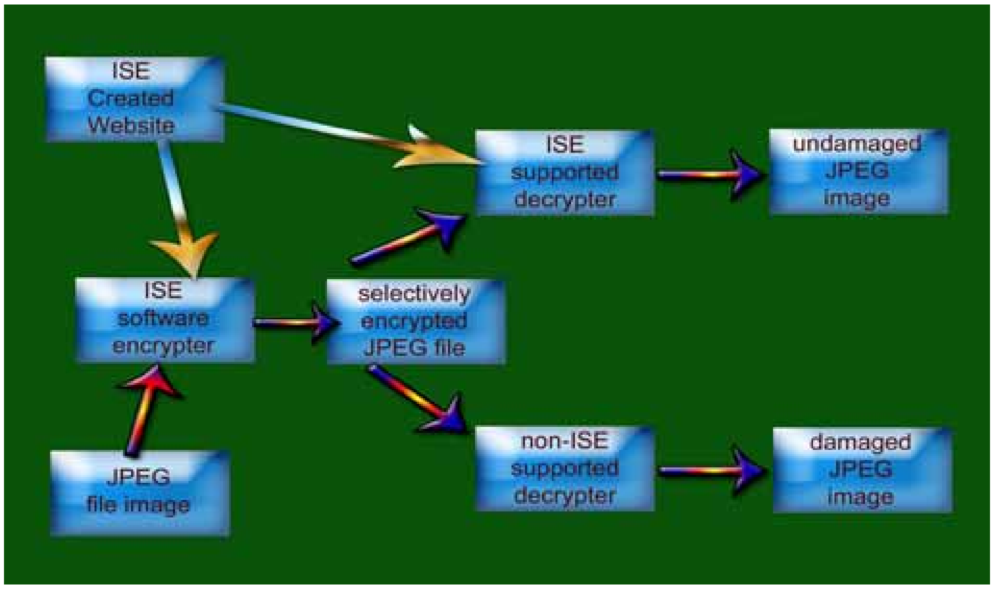


Figure 1.1: Conceptual Overview of ISE Software

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:

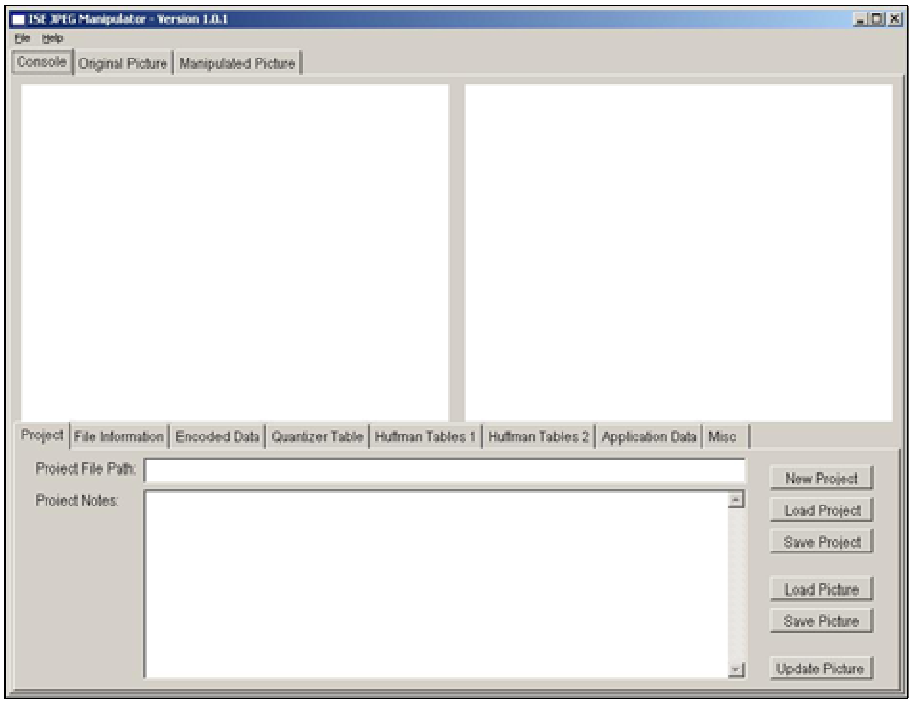


Figure 2.2.1: The JPEG Manipulator on entry into the application.

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.

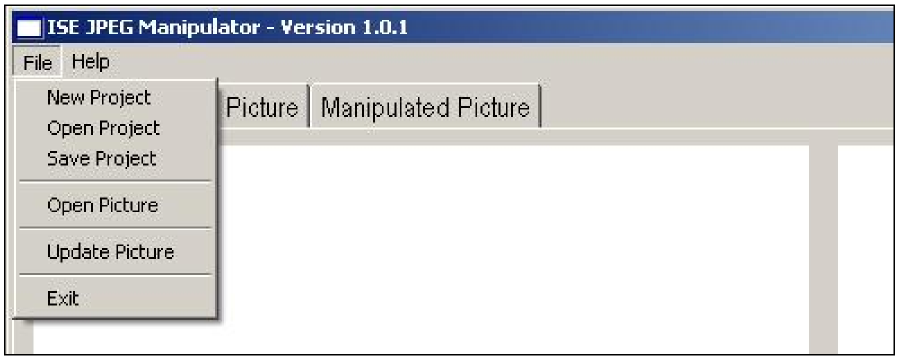


Figure 2.2.2.1: Illustration of the File menu.

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.

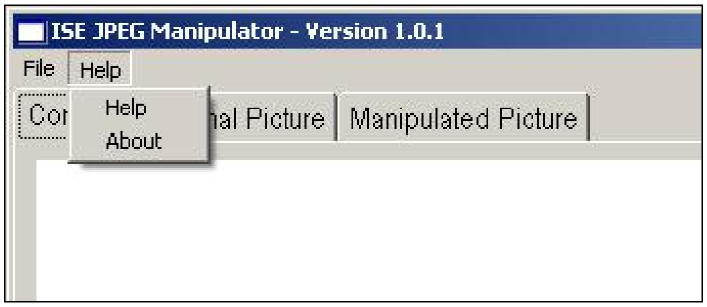


Figure 2.2.2.2: Illustration of the Help menu.

2.2.2.3.1. Help option

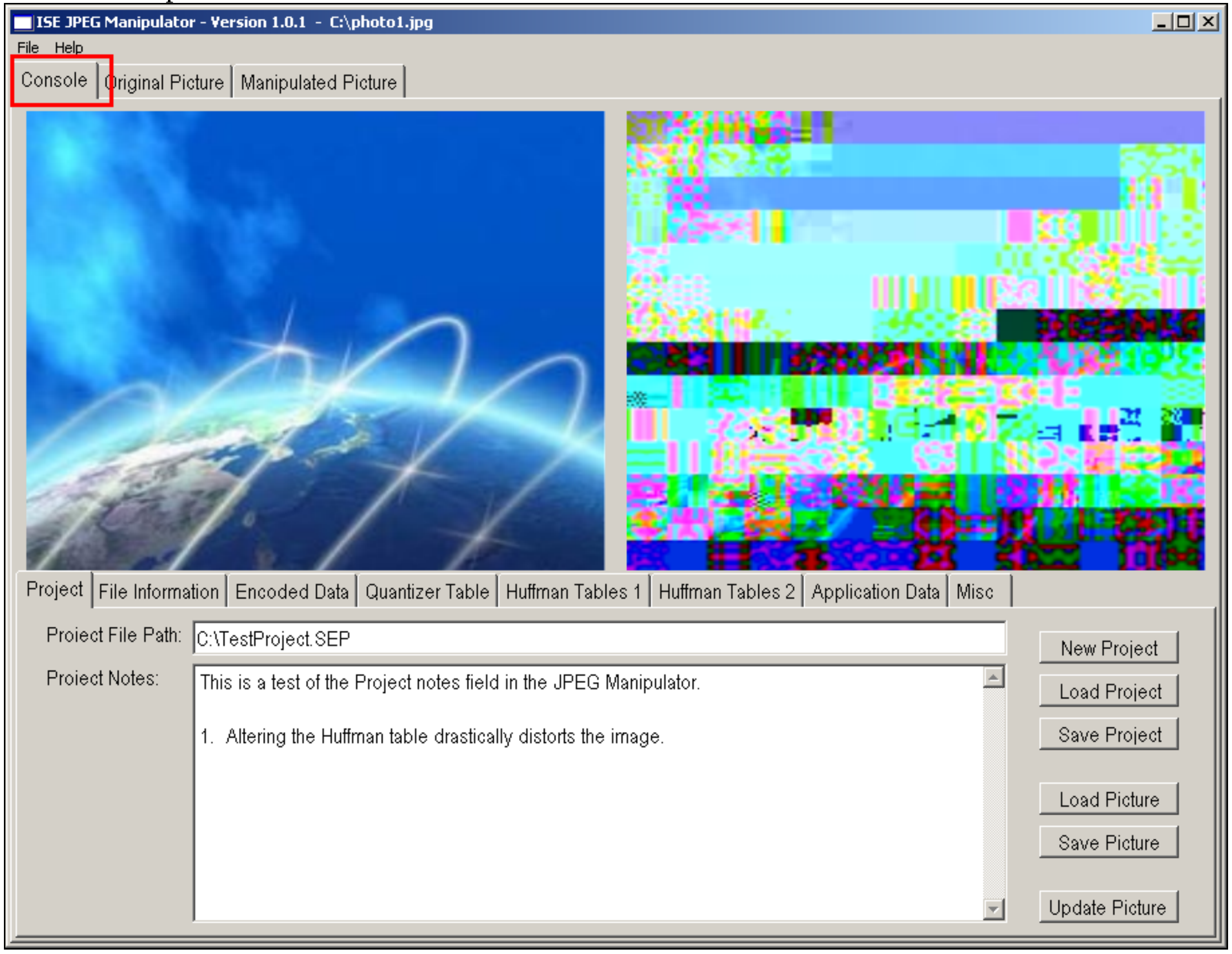
This option allows the user to enter the self-help portion of the program. This program will provide the user with a user-guide for the Manipulator and other information about using this program.

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:



Project File Path: C:\TestProject.SEP

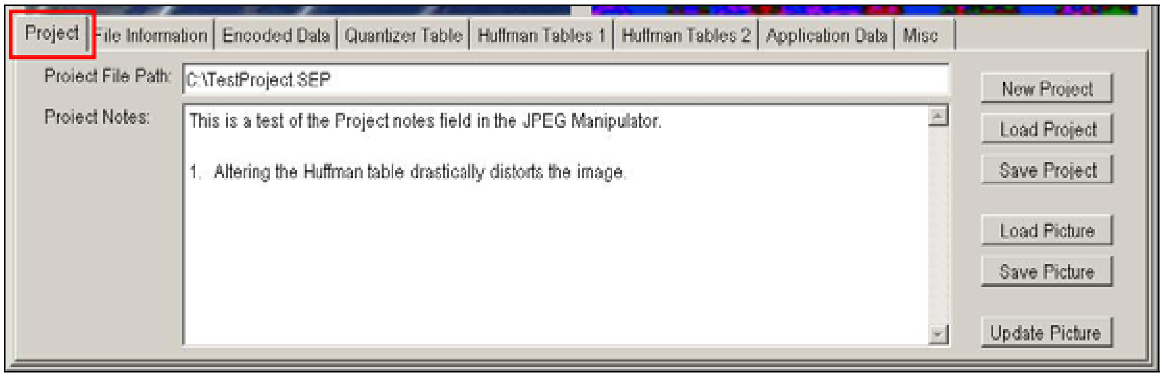
Project Notes: This is a test of the Project notes field in the JPEG Manipulator.

1. Altering the Huffman table drastically distorts the image.

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:



Project File Path: C:\TestProject.SEP

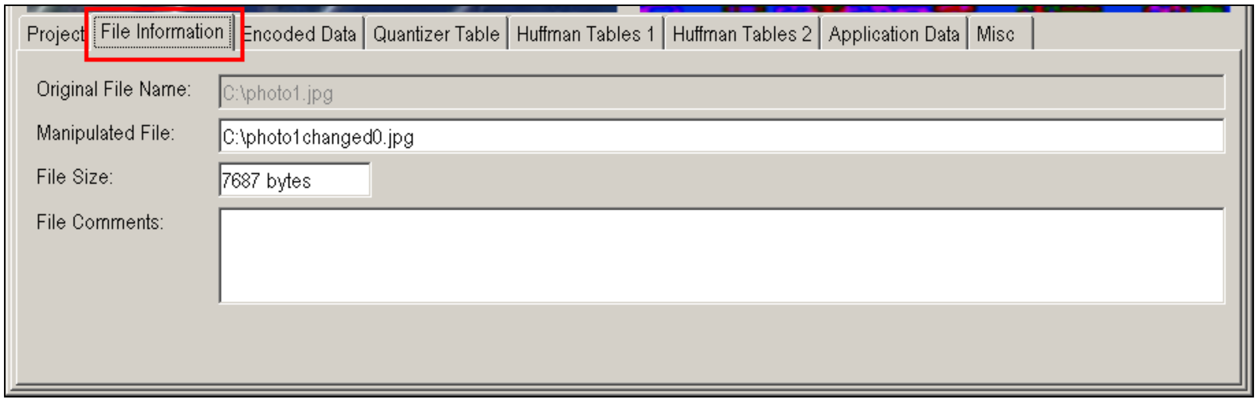
Project Notes: This is a test of the Project notes field in the JPEG Manipulator.

1. Altering the Huffman table drastically distorts the image.

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:



Original File Name: C:\photo1.jpg

Manipulated File: C:\photo1changed0.jpg

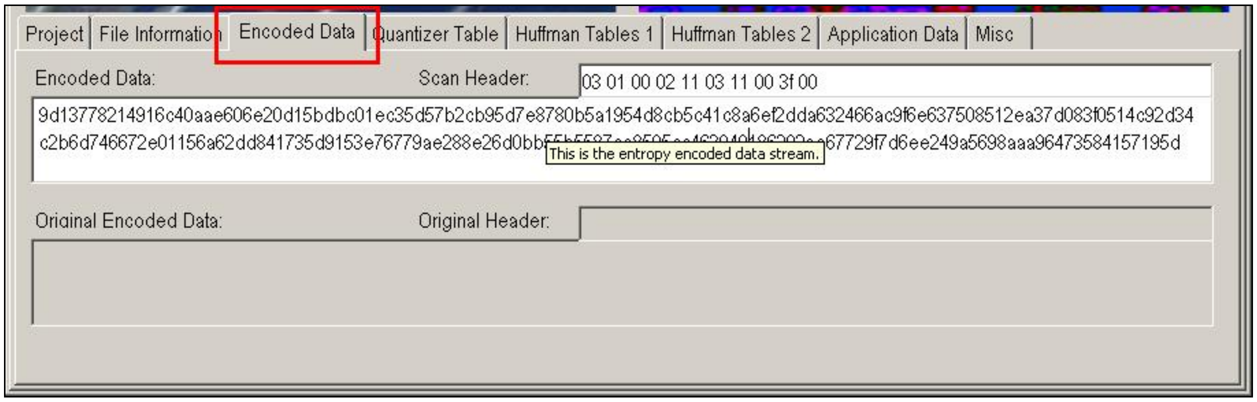
File Size: 7687 bytes

File Comments:

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:



Encoded Data:

```
9d13778214916c40aae606e20d15bdbc01ec35d57b2cb95d7e8780b5a1954d8cb5c41c8a6ef2dda632466ac9f6e637508512ea37d083f0514c92d34
c2b6d746672e01156a62dd841735d9153e76779ae288e26d0bb[illegible]67729f7d6ee249a5698aaa96473584157195d
```

This is the entropy encoded data stream.

Scan Header: 03 01 00 02 11 03 11 00 3f 00

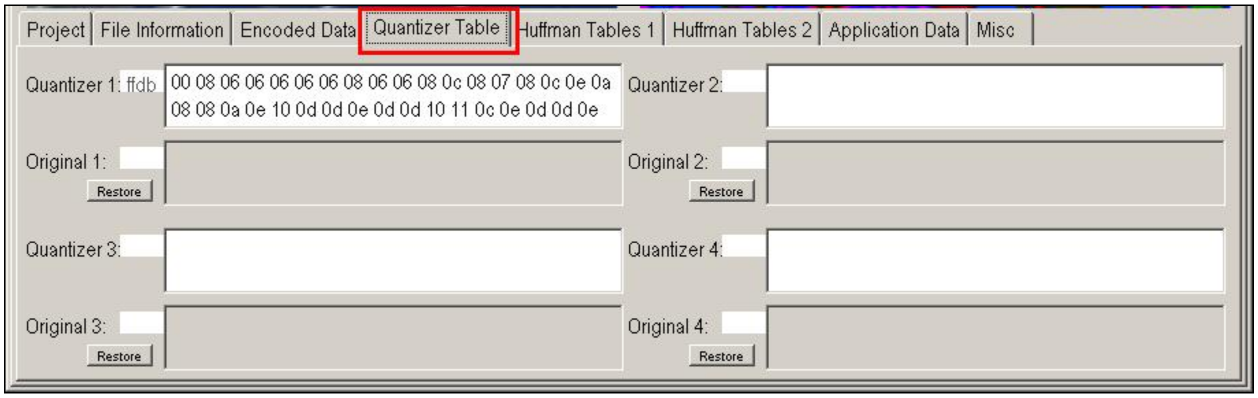
Original Encoded Data:

Original Header:

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:



Quantizer 1: ffdb

```
00 08 06 06 06 06 06 08 06 06 08 0c 08 07 08 0c 0e 0a
08 08 0a 0e 10 0d 0d 0e 0d 0d 10 11 0c 0e 0d 0d 0e
```

Quantizer 2:

Original 1:

Original 2:

Quantizer 3:

Quantizer 4:

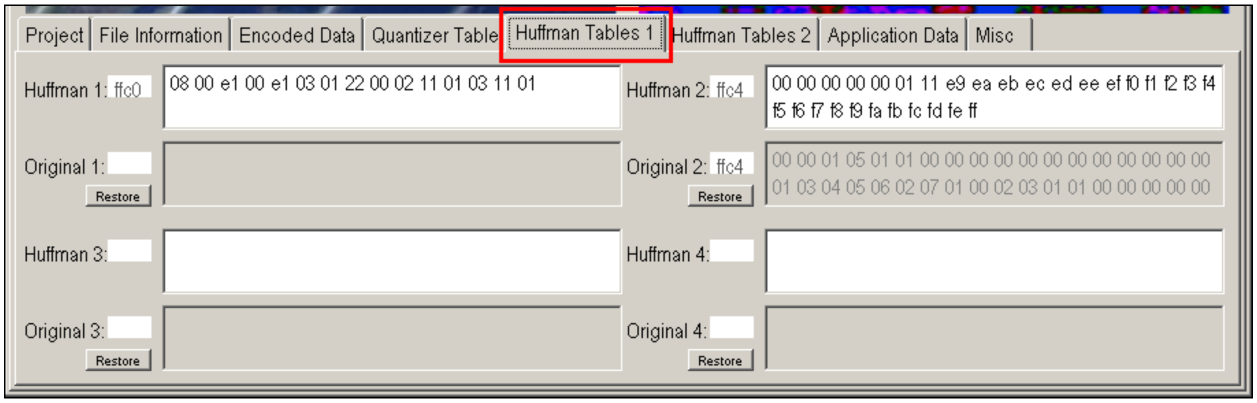
Original 3:

Original 4:

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:



Huffman 1: ffc0 08 00 e1 00 e1 03 01 22 00 02 11 01 03 11 01

Original 1:

Huffman 2: ffc4 00 00 00 00 00 01 11 e9 ea eb ec ed ee ef f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

Original 2: ffc4 00 00 01 05 01 01 00 00 00 00 00 00 00 00 00 00 00 00 01 03 04 05 06 02 07 01 00 02 03 01 01 00 00 00 00 00

Huffman 3:

Huffman 4:

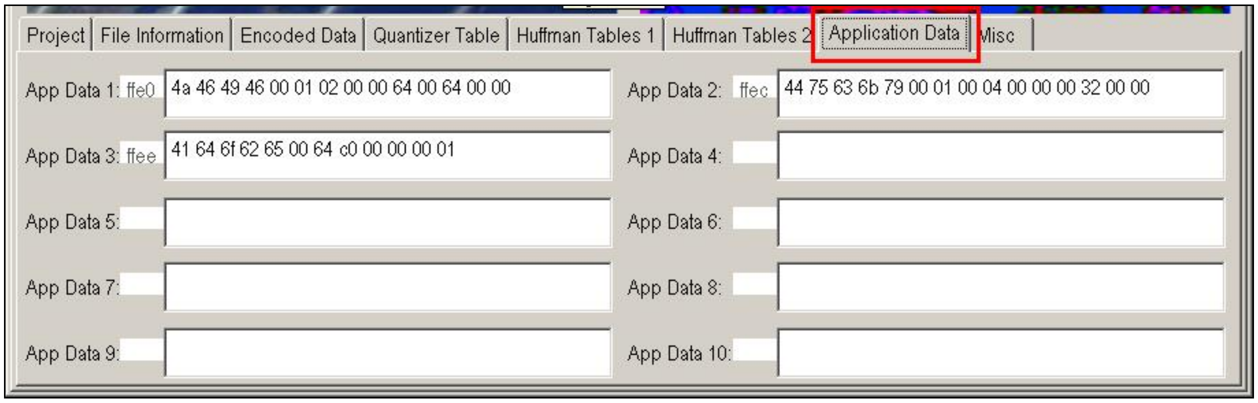
Original 3:

Original 4:

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:



App Data 1: ffe0 4a 46 49 46 00 01 02 00 00 64 00 64 00 00

App Data 2: ffec 44 75 63 6b 79 00 01 00 04 00 00 00 32 00 00

App Data 3: ffee 41 64 6f 62 65 00 64 c0 00 00 00 01

App Data 4:

App Data 5:

App Data 6:

App Data 7:

App Data 8:

App Data 9:

App Data 10:

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:

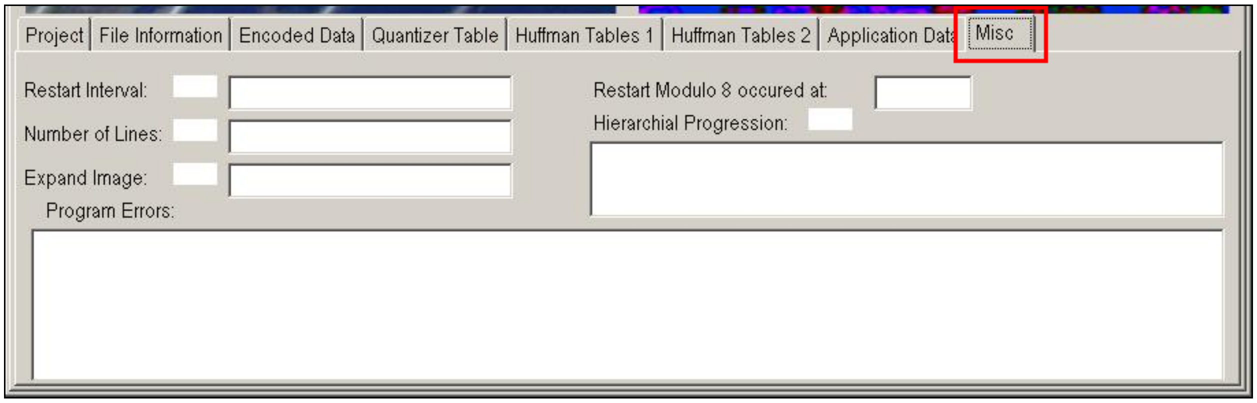


Figure 2.2.3.7: Example of the Misc Data Tab.

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:

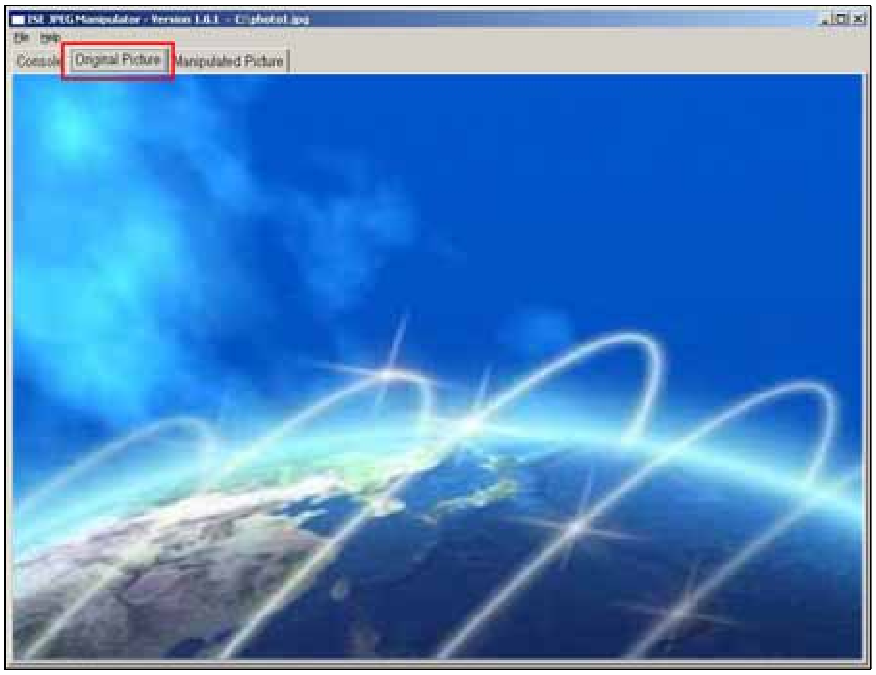


Figure 2.2.4: Example of the Original Picture Tab.

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:

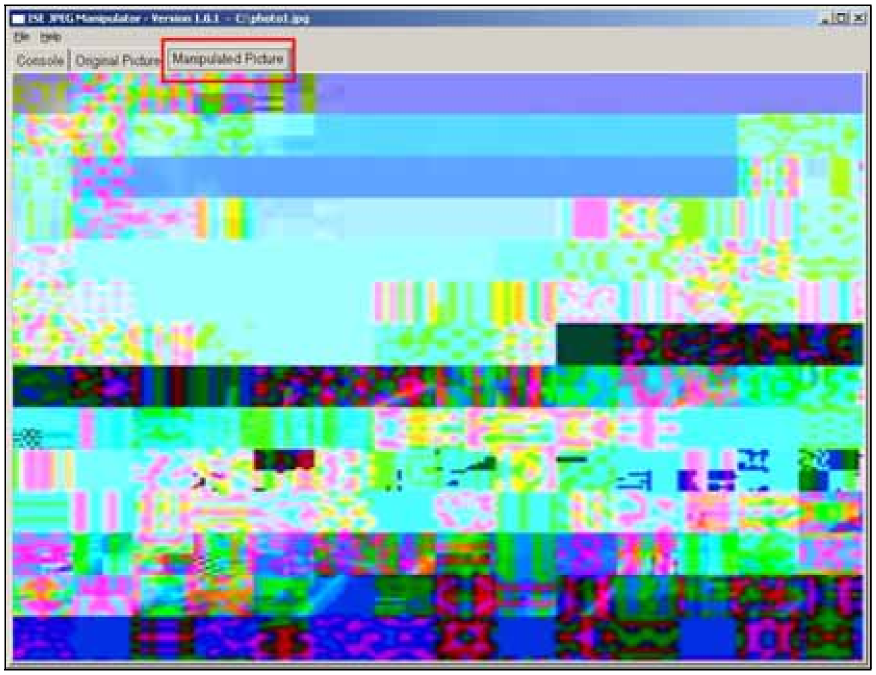


Figure 2.2.5: Example of the Manipulated Picture Tab.

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.

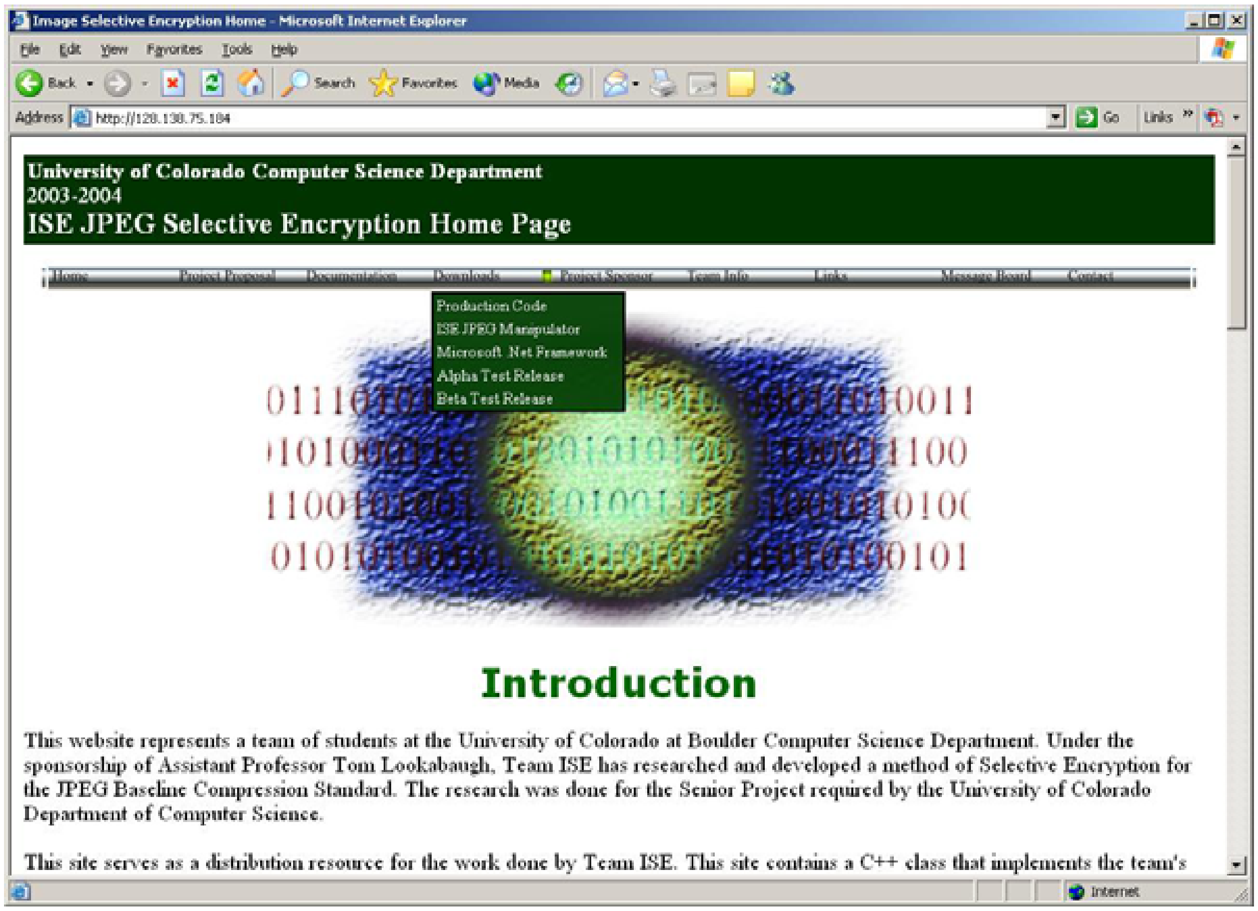
2.3.1. ISE Web Site Invocation

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.



University of Colorado Computer Science Department
2003-2004
ISE JPEG Selective Encryption Home Page

Introduction

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.

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

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.

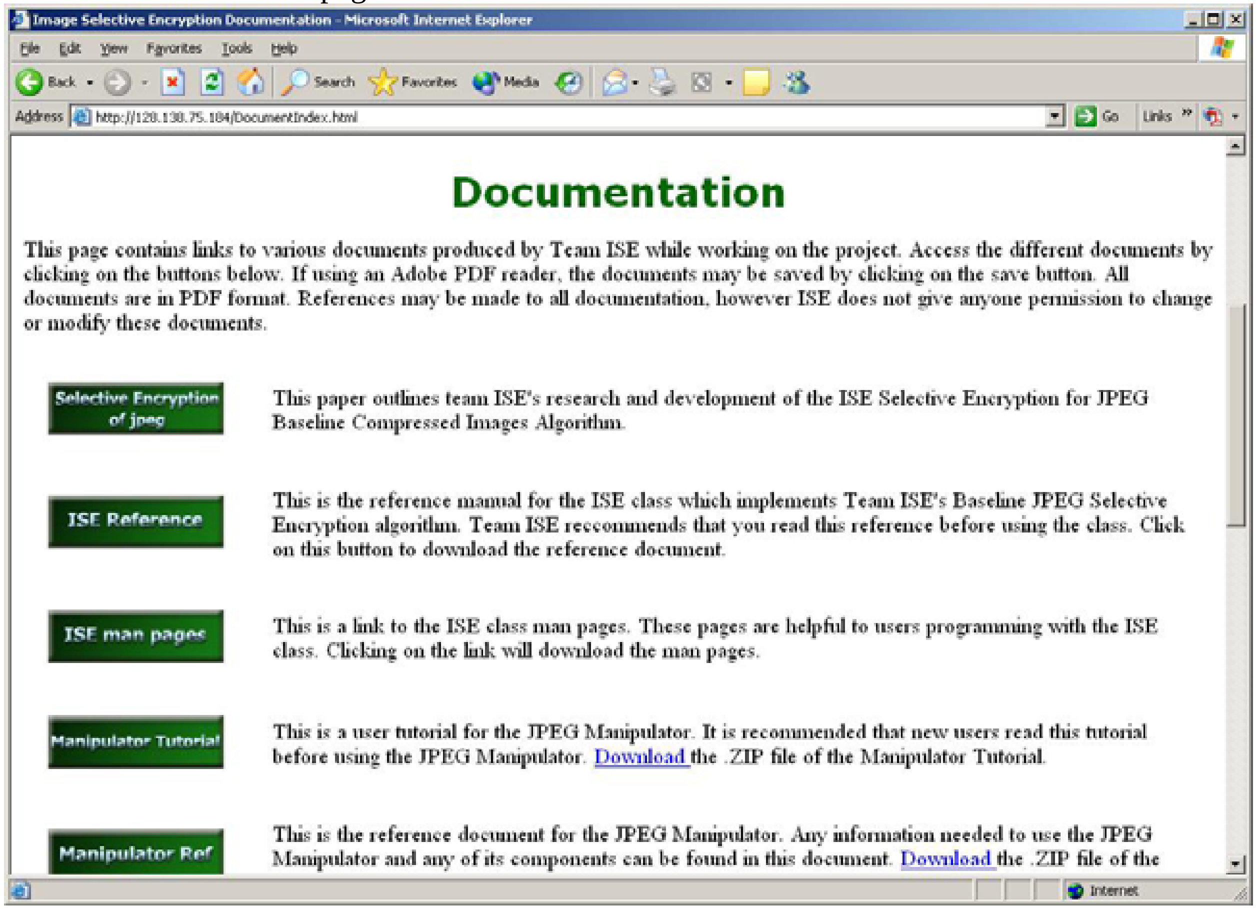


Image Selective Encryption Documentation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://128.138.75.184/DocumentIndex.html
Go Links
Internet

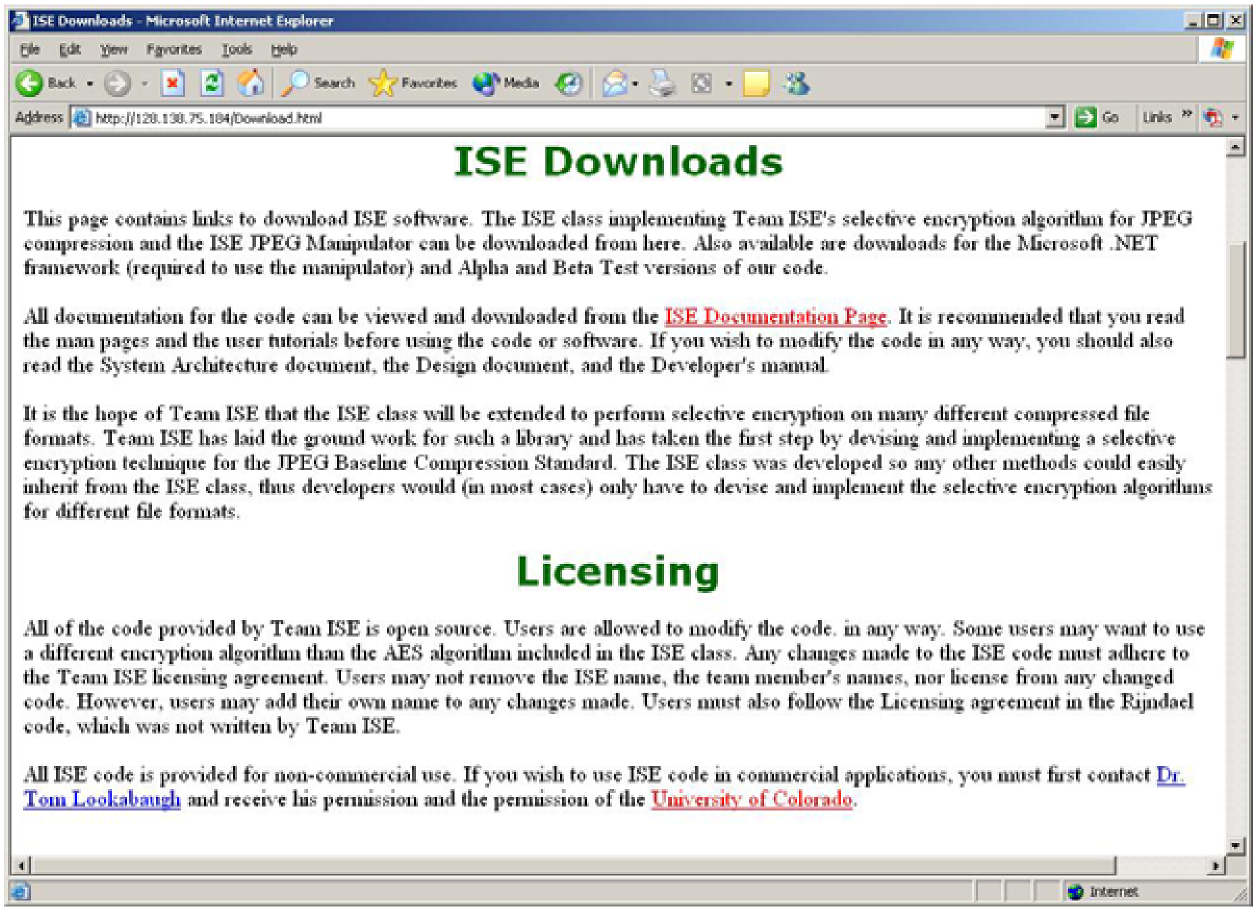
Documentation

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.

Selective Encryption of jpeg	This paper outlines team ISE's research and development of the ISE Selective Encryption for JPEG Baseline Compressed Images Algorithm.
ISE Reference	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.
ISE man pages	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.
Manipulator Tutorial	This is a user tutorial for the JPEG Manipulator. It is recommended that new users read this tutorial before using the JPEG Manipulator. Download the .ZIP file of the Manipulator Tutorial.
Manipulator Ref	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. Download the .ZIP file of the

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.


ISE Downloads - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://128.138.75.184/Download.html
Go Links
Internet

ISE Downloads

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.

All documentation for the code can be viewed and downloaded from the ISE Documentation Page. 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.

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.

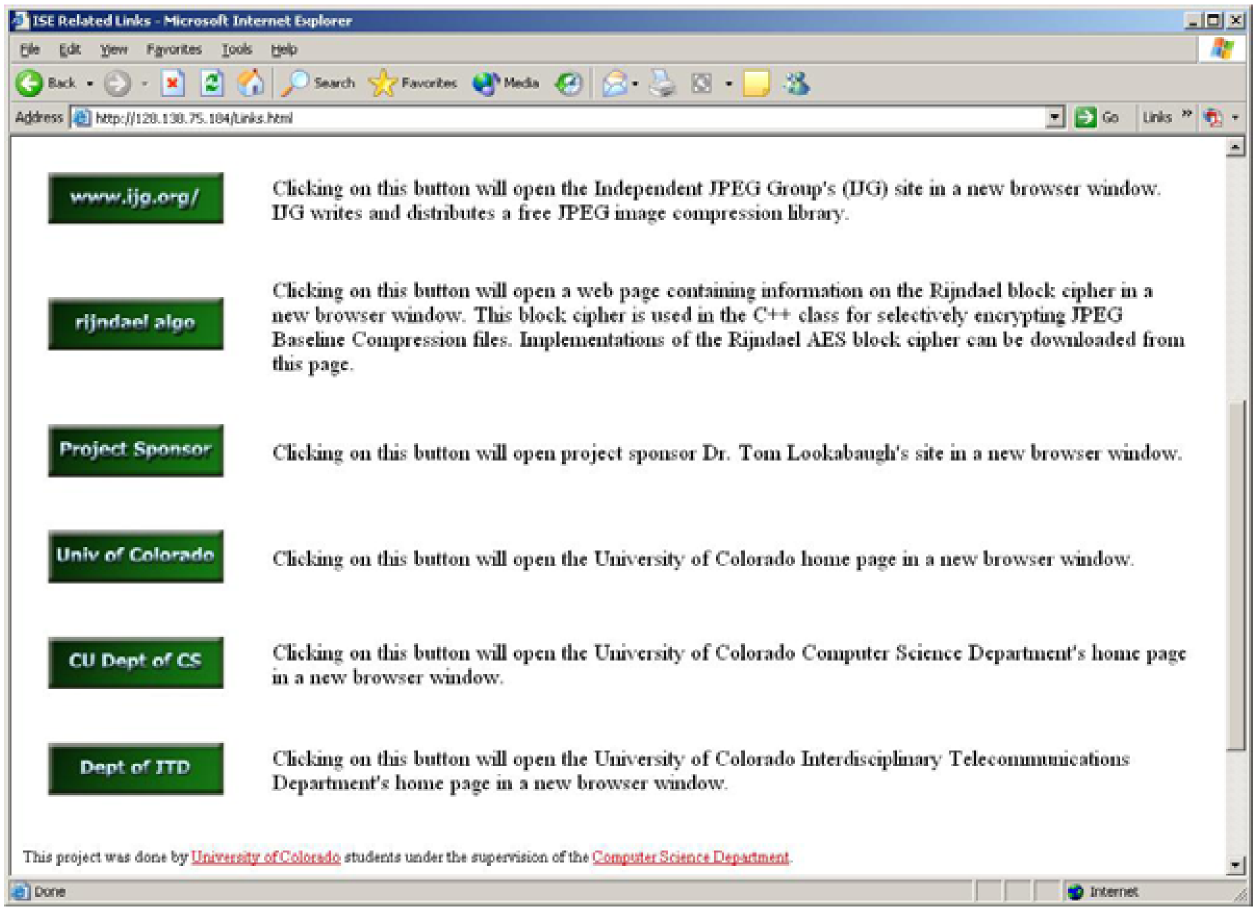
Licensing

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.

All ISE code is provided for non-commercial use. If you wish to use ISE code in commercial applications, you must first contact Dr. Tom Lookabaugh and receive his permission and the permission of the University of Colorado.

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.


ISE Related Links - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://128.138.75.184/Links.html
Go Links
www.ijg.org/
rijndael algo
Project Sponsor
Univ of Colorado
CU Dept of CS
Dept of ITD
Done
Internet

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.

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.

Clicking on this button will open project sponsor Dr. Tom Lookabaugh's site in a new browser window.

Clicking on this button will open the University of Colorado home page in a new browser window.

Clicking on this button will open the University of Colorado Computer Science Department's home page in a new browser window.

Clicking on this button will open the University of Colorado Interdisciplinary Telecommunications Department's home page in a new browser window.

This project was done by University of Colorado students under the supervision of the Computer Science Department.

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:

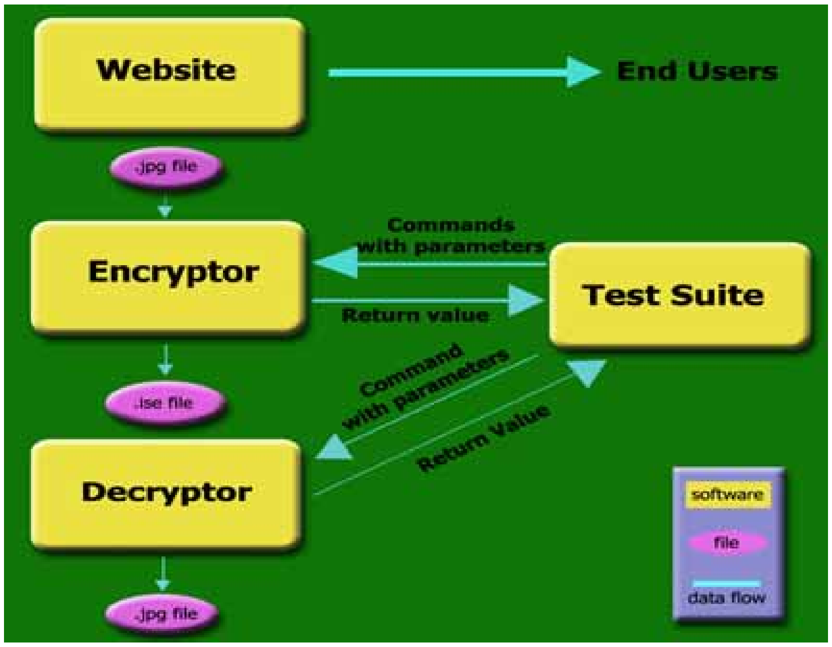


Figure 3.1: High level modular decomposition of the ISE project.

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.