P - 1616

DIGITAL IMAGE WATERMARKING WITH HARDWARE KEY

A PROJECT REPORT

Submitted by

N. ARUN KUMAR (71202104003)

K.M.DILIP (71202104059)

A.RAMAKRISHNAN (71202104061)

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641006

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2006

ANNAUNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**Digital Image Watermarking with Hardware Key**" is the bonfire work of "**Arunkumar.N, Dilip.K.M, Ramakrishnan.A**" who carried out the project work under my supervision.

[illegible]

SIGNATURE
Dr. S. THANGASAMY
HEAD OF THE DEPARTMENT

Computer Science & Engineering
Kumaraguru College of Technology
Chinnavedampatti (p-o)
Coimbatore-641006

[illegible]

SIGNATURE
D.CHANDRAKALA
SUPERVISOR
ASSISTANT PROFESSOR
Computer Science & Engineering
Kumaraguru College of Technology
Chinnavedampatti (p-o)
Coimbatore - 641006

Submitted for viva-voce examination held on 2/5/06

[illegible]
Internal Examiner

External Examiner

DECLARATION

We here by declare that the project entitled "Digital Image Watermarking with Hardware Key", is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any other institution for fulfillment of the requirement of the course study

This report is submitted in partial fulfillment of the requirements for the award of the degree of bachelor of computer science and engineering of Anna University, Chennai

[illegible]

Place: Coimbatore (N.Arun Kumar)

Date: 20-04-2006

[illegible]

(K.M.Dilip)

[illegible]

(A.Ramakrishnan)

ACKNOWLEDGEMENT

I would like to express my heartfelt thanks to **Dr.K.K.Padmanaban, Principal of Kumaraguru College of Technology**, Coimbatore, for having permitted me to do this project.

I thank **Dr. S.Thangasamy, Department of Computer Science and Engineering**, for providing me an opportunity to take up the project and for his continuous encouragement.

I express my sincere my gratitude to **Mrs. P. Devaki, Project Coordinator**, for her valuable suggestions and valuable guidance throughout project.

I am indebted to thank **Mrs. D.Chandrakala, Assistant professor, Computer science & Engineering Department** for her valuable guidance given throughout this project.

I wish to convey my sincere thanks to my beloved **Parents** who gave me the moral support in making this project a grand success.

Last but not the least, I express my gratitude to all my **friends** who brought in the novel concept of "Knowledge Sharing" without which it would not have been possible for me to have learnt more in this pace.

ABSTRACT

A watermark is a secondary image which is overlaid on the primary image, and provides a means of protecting the image.

The Objective of the project is to avoid the hacking of the company picture and provide security to the picture. Java and Visual Basic are the languages used to develop this project.

The Major Modules of the Project are

1. Hardware Module
2. Software Module

This project helps to protect the pictures by using the company Logo. This Project gives Security to Pictures and the Hardware Key gives Security to the Software. The watermarked image created by this Software cannot be removed, altered or damaged. It is used to protect documents, Pictures and emblems. The watermarked picture gives the authority or proof of our ownership.

Trail version will have only the limited features. The picture cannot be saved in this version. In the trial version when we watermark a picture the text "demo" will be added with the watermarking picture. The full version can be accessed by using the hardware key. The hardware key should be connected in the serial port to access full version of the software and if we remove the hardware key from the serial port we can use only the trial version of software. In the full version of this project, we can save the watermarked image can be saved in some particular format.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

COM	- Component Object Model
DFD	- Data Flow Diagram
VB	- Visual Basic
AC	- Alternative Current
DC	- Direct Current
V	- Voltage in Volts
IC	- Integrated Circuit
TTL	- Transistor – Transistor Logic
RS	- Recommended Standard
SIC	- Standard Industrial Classification
FIFO	- First in first out
CPU	- Central Processing Unit
UART	- Universal asynchronous receiver and transmitter
RXD	- Received Data
TXD	- Transmitted Data
PCON	- Power control
SCON	- Serial control
TMOD	- Timer Mode
SBUF	- Serial Buffer
AWT	- Abstract Windowing Toolkit

INTRODUCTION

1. INTRODUCTION

1.1 PROBLEM DEFINITION

The development of compression technology as JPEG allowed the wide-spread use of multimedia applications. Nowadays, digital documents can be distributed via the World Wide Web to a large number of people in a cost-efficient way. The increasing importance of digital media, however, brings also new challenges as it is now straightforward to duplicate and even manipulate multimedia content. There is a strong need for security services in order to keep the distribution of digital multimedia work both profitable for the document owner and reliable for the customer. Digital Image Watermarking technology plays an important role in securing the images. The purpose of digital watermarks is to provide copyright protection for intellectual property that's in digital format. The software protects the images from copying and redistribution. It is not possible to remove, alter or damage the watermarked images.

1.2 EXISTING SYSTEM

The Existing system doesn't give picture protection. It also affects the quality of the watermarked image. The software is easily pirated by the user. The software key of the existing system is also pirated by the users. So there is less security for both the picture and software.

The watermarked picture of the existing system is edited easily by the users. The system embeds the secondary image over the primary image. The Quality of the original contents is not same when compared to the watermarked image quality.

The RGB Colors also affected, while we create a water marked image through existing system. It could affect the quality of the Picture. The transparency of the secondary image in the watermarked image also affects the visibility of the primary image.

1.3 PROPOSED SYSTEM

The proposed system should provide efficient usage of the project for the customer. The project is developed for reducing the software piracy. Our proposed system solves the drawback of existing system by means of security for both picture and software.

The main concern of system is to protect the image used on the webpage in the internet from coping and redistribution .The imgea are watermarked so that it is not possible to remove,alter or damage those images.

There is a strong need for security services in order to keep the distribution of digital multimedia work both profitable for the document owner and reliable for the customer. Digital Image Watermarking technology plays an important role in securing the images. The purpose of digital watermarks is to provide copyright protection for intellectual property that's in digital format The software protects the images from copying and redistribution.

While comparing to the existing software in the market, this project gives security for both the software and the picture. Hardware Key gives Security for Software. The watermarked image created by this Software cannot be removed, altered or damaged.

Watermarked content has the same subjective quality as the original contents. In trail Version, We can only see the quality of the software. There is no saving option in the trail version. We can save the watermarked image only in full version.

ANALYSIS

2. ANALYSIS

2.1 SCOPE OF SYSTEM

The system is developed by taking into consideration, the protection of images used on the web pages in the Internet. It protects the images from copying and redistribution. A watermark is a secondary image which is overlaid on the primary image and provides a means for protecting the image. It is not possible to remove, alter or damage the watermarked images.

The system is developed in two versions, the trial and full versions. The trial version has its own limitations like the pictures can be saved only in the fixed formats. While in the full version, digital watermarked images can be stored in the desired formats, watermarking picture or text can be placed anywhere in the image and the images can be opened, edited and saved. The full version can be accessed using the hardware key. Multiple picture watermarking can be done at the same time. The hardware key prevents piracy and thereby highly secure than the serial key.

2.2 SYSTEM ENVIRONMENT

2.2.1 HARDWARE PLATFORM

CPU	: Pentium IV
RAM	: 256 MB
Hard Disk	: 40 GB
Floppy Drive	: 1.44 MB
CD Drive	: 52X
Monitor	: Plug and Play
Mouse	: Logitech Scroll mouse
Keyboard	: 120 Keys
IC	: 89C51
IC Programmer	: SUPERPRO

2.2.2 S/W PLATFORM

Operating System	: Windows XP
Programming Language	: Java
Microcontroller Programming	: Assembly Language
Interfacing	: Visual Basic
Components	: COM

2.3 SOFTWARE PROFILE

2.3.1 JAVA

MAIN

The main method can be put in any class, and you will find a lot of programs that just stick it in the JFrame subclass. This style doesn't create many problems, so it isn't much of a sin if you do it this way. However, I've chosen another common way -- put it in a separate class.

- Most beginners understand this more easily, and feel uncomfortable when main creates an object of its own class. Moving main to its own class is simple, and makes the code easier to understand.
- Separating main makes it easier to change between an application and an applet without confusion. It's possible to put main in a JApplet subclass, but again it leads to confusion with no special advantage.
- It moves main's unchanging window organization code to its own place, away from other code so there is less confusion. When main is in the same class as the GUI, there is a temptation to try to reference instance variables from main, which leads to major headaches.
- For simple programs a small main looks lonely in its own class, but as programs grow larger, main will take on more initialization and framework responsibilities, so putting it in its own class gets you used to a style that will be useful when your programs get larger.

DECLARATION

Some components will be created in the instance variable declaration area, but components that are stored in local variables will be created in the constructor. A slight inconsistency, which may be irrelevant. If they are all initialized in the constructor, then there's only one place to look for initializations.

Even if a component is created in the declaration, it is not uncommon to set additional attributes, which must be done in the constructor. This splits the code for the component into two places.

The instance variables are typically the "most interesting" because they are used in multiple places. If they are all initialized at their point of declaration, it's easy to find all of these variables and their initial values.

WINDOW LISTENER

When a window of a program is closed, you typically want to check to see if there is any unsaved data, and if so, ask the user if they want to save it. To do this you need to add a window listener.

Although adding a window listener is the right way to handle the close box click in a real program, simple programs often don't have any state that needs to be saved, so I've used the common shortcut of terminating the program when the window is closed. The default action of closing a window just makes it invisible, but the program continues to run, making it hard to stop the program, so it's essential to have either this or a listener.

INTERFACES

More important than inheritance are ***interfaces***. Again, it is hard to create small, real, examples of interface usage that can be related to real student programs at a fairly early stage. Primarily interfacing is in the form of ActionListener, is a good model for interfaces at an early stage.

AWT

Java's first attempt was called the Abstract Windowing Toolkit (AWT). which was based on the underlying native system components. Some AWT classes, not including components, are still commonly used, which is why you typically have to import from both java.awt and javax.swing

LAYOUTS

Layout managers eliminate the need to compute component placement on your own, which would be a losing proposition since the size required for any component depends on the platform on which it is displayed.

BORDER LAYOUT

Border Layout is the default LayoutManager for a Window. It provides a very flexible way of positioning components along the edges of the window. The following call to setLayout() changes the LayoutManager of the current container to the default BorderLayout: setLayout(new BorderLayout()).

Border Layout is the only layout provided that requires you to name components when you add them to the layout; if you're using a Border Layout, you must use add(String name, Component

component) in Java 1.0 or add(Component component, String name) in Java 1.1 (parameter order switched). (The CardLayout can use these versions of add(), but does not require it.) The name parameter of add() specifies the region to which the component should be added. The five different regions are "North", "South", "East", and "West" for the edges of the window, and "Center" for any remaining interior space. These names are case sensitive. It is not necessary that a container use all five regions. If a region is not used, it relinquishes its space to the regions around it. If you add() multiple objects to a single region, the layout manager only displays the last one. If you want to display multiple objects within a region, group them within a Panel first, then add() the Panel.

public static final String CENTER

The CENTER constant represents the "Center" string and indicates that a component should be added to the center region.

public static final String EAST

The EAST constant represents the "East" string and indicates that a component should be added to the east region.

public static final String NORTH

The NORTH constant represents the "North" string and indicates that a component should be added to the north region.

public static final String SOUTH

The SOUTH constant represents the "South" string and indicates that a component should be added to the south region.

public static final String WEST

The WEST constant represents the "West" string and indicates that a component should be added to the west region.

public BorderLayout ()

This constructor creates a BorderLayout using a default setting of zero pixels for the horizontal and vertical gaps. The gap specifies the space between adjacent components. With horizontal and vertical gaps of zero, components in adjacent regions will touch each other. As Figure 7-4 shows, each component within a BorderLayout will be resized to fill an entire region.

public BorderLayout (int hgap, int vgap)

This version of the constructor allows you to create a BorderLayout with a horizontal gap of hgap and vertical gap of vgap, putting some space between the different components. The units for gaps are pixels. It is possible to have negative gaps if you want components to overlap.

GRID LAYOUT

The Grid Layout layout manager is ideal for laying out objects in rows and columns, where each cell in the layout has the same size. Components are added to the layout from left to right, top to bottom. Set Layout(new GridLayout(2,3)) changes the LayoutManager of the current container to a 2 row by 3 column Grid-Layout.

public GridLayout ()

This constructor creates a GridLayout initially configured to have one row, an infinite number of columns, and no gaps. A gap is the space between adjacent components in the horizontal or vertical direction. With a gap of zero, components in adjacent cells will have no space between them.

public GridLayout (int rows, int columns)

This constructor creates a GridLayout initially configured to be rows columns in size. The default setting for horizontal and vertical gaps is zero pixels. The gap is the space between adjacent components in the horizontal and vertical directions. With a gap of zero, components in adjacent cells will have no space between them. You can set the number of rows or columns to zero; this means that the layout will grow without bounds in that direction. If both rows and columns are zero, the run-time exception Illegal Argument Exception will be thrown.

public GridLayout (int rows, int columns, int hgap, int vgap)

This version of the constructor is called by the previous one. It creates a Grid-Layout with an initial configuration of rows columns, with a horizontal gap of hgap and vertical gap of vgap. The gap is the space between the different components in the different directions, measured in pixels. It is possible to have negative gaps if you want components to overlap.You can set the number of rows or columns to zero; this means that the layout will grow without bounds in that direction. If both rows and columns are zero, the run-time exception Illegal Argument Exception will be thrown.

SWING

Swing did not replace everything in AWT, therefore you will use a mixture of the two. However, never mix the components from Swing and AWT. The Swing component names generally start with 'J' (eg JButton), and the AWT components don't (eg Button). Although mixing the two types will not produce a compile-time error, the result may not display correctly. The Graphics, Color, Font, layout, listener, ... classes are from AWT, so you typically need to have imports from AWT

It supports the Java Look and Feel interface which allows the interface to be rendered very close to a native look, although not pixel-perfect. This is the mainstream choice.

JFrames (and subclasses of them) have a *content pane*, which holds the components, which are positioned by the layout manager when pack() is called. There are no special problems with this, so if you are just interested in getting your program running, stop reading now. What follows is information that will help you understand variations you'll see in other programs.

Every JFrame comes with a content pane which has already been created. Here are some reasons the examples create a new JPanel instead of using the preconstructed content pane. These don't form a strong argument against using the predefined pane, but you'll see the source of my preference. The overall layout of a window is often created by nesting panels. If you're going to be creating panels and working with them, it's easier to treat them all the same rather than making the top-level content pane a special case.

getContentPane() returns a Container object. This isn't really a plain Container object, but is actually a JPanel, which is a subclass of

JComponent, which is a subclass of Container! So if we get the predefined content pane, it turns out it's actually a JPanel, but we really can't take advantage of the functionality that was added to a plain container by JComponent and JPanel. If we downcast it to JPanel, we create fragile code that might break because the contract with getContentPane() is to return a Container, and there is no guarantee that future versions will actually continue to return a JPanel. So if you use the returned Container you can't take advantage of useful JPanel features like setBorder().

Before the Swing GUI library, there was no content pane and components were added directly to the window (the old AWT Frame class). This seemed easy and it was annoying to rewrite code to use the content pane. But that was long ago, and for most of Java's history programmers have been using the content pane. For some reason, Java 5 added the ability to add directly to the JFrame again -- probably because it's been on some list of suggestions for a really long time, and it was trivial to implement. They defined add() methods in JFrame which simply call the corresponding add() methods for the content pane. It seems bizarrely out-of-touch to add this feature now, especially since most layouts have multiple nested panels so you still have to be comfortable with adding directly to JPanels.

There is no problem if you want to use it, but why bother learning a minimally useful alternative when you have to know how to add to a JPanel. Especially when they refuse to add useful features because they are concerned about the every growing size of the Java API.

2.3.2 Visual Basic

Visual Basic is used for interfacing the Hardware Key with Software. We using COM object in this project for interfacing.

COM - Component Object Model

- Com is a specification for a way of creating components and building applications from these Components.

- Com was developed four years ago to make applications more flexible, dynamic and customizable.

- E.g. :- Microsoft's Active x Technologies are built from COM components

Component is nothing but converting ordinary objects into persistence

Objects by means of

a) Properties

b) Event

c) Introspection

d) Customization

e) Persistence or Consistence

Functionality of Components

First, we must understand how the component works and how its functionality differs from the application.

We can analyze it in 3 ways

1. Analyzing Application Functionality
2. Component Reusability
3. Expert Functionality

Analyze Application Functionality

One of the main characteristic of a component is that the business logic is separate from the data. While creating a component it is important to manipulate data in arbitrary collections or streams.

Component Reusability:

The component produced should be of commercial viability. Viability means able to continue to live under altered conditions.

Expert Functionality:

While writing component we should focus on Expertise and knowledge. If the components developed from a scratch than we should access whether we can provide superior solution.

2.3.3 Assembly Language

Assembly language is used to program in the microcontroller. We can see the expansion of the commands used in this project

- Equ – equate a value to a particular variable
- Org – move to a particular address that is the origin
- Mov – move the address or value to a particular memory location or register
- Anl – AND logic with the value and register
- Orl – OR logic with the value and register
- Setb – set 1 to the variable
- Acall – call to subroutine
- Sjmp – short jump to a particular memory location
- Nop – No operation
- Djnz – Decrement jump to a memory location while the variable is not zero
- Ret – return
- Pcon – Power control
- Scon – Serial control
- Tmod – Timer Mode
- Tr1 – Timer1

2.4 Hardware Profile

2.4.1 IC 89C51 Microcontroller

Ports 0 To 3

P0, P1, P2, and P3 are the SFR latches of Ports 0, 1, 2, and 3, respectively.

Serial Data Buffer (SBUF)

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer, where it is held for serial transmission. (Moving a byte to SBUF initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

Timer Registers

Register pairs (TH0, TL0) and (TH1, TL1) are the 16-bit Counter registers for Timer/Counters 0 and 1, respectively.

Control Registers

Special Function Registers TMOD, TCON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port.

TIMER/COUNTERS

The IC89C51 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. All two can be configured to operate either as Timers or event Counters. As a Timer, the register is incremented every machine cycle. Thus the register counts machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency. As a Counter, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 and T1. The external input is sampled during S5P2 of every machine cycle.

When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but it should be held for at least one full machine cycle to ensure that a given level is sampled at least once before it changes. In addition to the Timer or Counter functions, Timer 0 and Timer 1 have four operating modes: 13-bit timer, 16-bit timer, 8-bit auto-reload, split timer.

Timer 0 and Timer 1

The Timer or Counter function is selected by control bits C/T in the Special Function Register TMOD. These two Timer/Counters have four operating modes, which are selected by bit pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timer/Counters, but Mode 3 is different. The four modes are described in the following sections.

Mode 0:

Both Timers in Mode 0 are 8-bit Counters with a divide-by-32 prescaler. It shows the Mode 0 operation as it applies to Timer 1.

In this mode, the Timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or INT1 = 1. Setting GATE = 1 allows the Timer to be controlled by external input INT1, to facilitate pulse width measurements. TR1 is a control bit in the Special Function Register TCON. Gate is in TMOD.

The 13-bit register consists of all eight bits of TH1 and the lower five bits of TL1. The upper three bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

Mode 0 operation is the same for Timer 0 as for Timer 1, except that TR0, TF0 and INT0 replace the corresponding Timer 1 signals in Figure 8. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

Mode 1:

Mode 1 is the same as Mode 0, except that the Timer register is run with all 16 bits. The clock is applied to the combined high and low timer registers (TL1/TH1). As clock pulses are received, the timer counts up: 0000H, 0001H, 0002H, etc. An overflow occurs on the FFFFH-to-0000H overflow flag. The timer continues to count. The overflow flag is the TF1 bit in TCON that is read or written by software.

Mode 2:

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is preset by software. The reload leaves the TH1 unchanged. Mode 2 operation is the same for Timer/Counter 0.

Mode 3:

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0. Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 11. TL0 uses the Timer 0 control bits: C/T, GATE, TR0, INT0, and TF0. TH0 is locked into a timer function (counting machine cycles) and over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the Timer 1 interrupt. Mode 3 is for applications requiring an extra 8-bit timer or counter. With Timer 0 in Mode 3, the IS89C51 can appear to have three Timer/Counters. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3. In this case, Timer 1 can still be used by the serial port as a baud rate generator or in any application not requiring an interrupt.

2.4.2 SERIAL INTERFACE

The Serial port is full duplex, which means it can transmit and receive simultaneously. It is also receive-buffered, which means it can begin receiving a second byte before a previously received byte has been read from the receive register. (However, if the first byte still has not been read when reception of the second byte is complete, one of the bytes will be lost.) The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

The serial port can operate in the following four modes:

Mode 0:

Serial data enters and exits through RXD. TXD outputs the shift clock. Eight data bits are transmitted/received, with the LSB first. The baud rate is fixed at 1/12 the oscillator frequency .

Mode 1:

Ten bits are transmitted (through TXD) or received (through RXD): a start bit (0), eight data bits, and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable .

Mode 2:

Eleven bits are transmitted (through TXD) or received (through RXD): a start bit (0), eight data bits, a programmable ninth data bit, and a stop bit (1). On transmit, the ninth data bit can be assigned the value of 0 or 1. Or, for example, the parity bit can be moved into TB8. On receive; the ninth data bit goes into RB8 in Special Function Register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency.

Mode 3:

Eleven bits are transmitted (through TXD) or received (through RXD): a start bit (0), eight data bits, a programmable ninth data bit, and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except the baud rate, which is variable in Mode 3. In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN =1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

2.4.3 RS 232

Serial RS-232 (V.24) communication works with voltages (-15V ... -3V for *high* [sic]) and +3V ... +15V for *low* [sic]) which are not compatible with normal computer logic voltages. On the other hand, classic TTL computer logic operates between 0V ... +5V (roughly 0V ... +0.8V for *low*, +2V ... +5V for *high*). Modern low-power logic operates in the range of 0V ... +3.3V or even lower.

So, the maximum RS-232 signal levels are far too high for computer logic electronics, and the negative RS-232 voltage for *high* can't be grokked at all by computer logic. Therefore, to receive serial data from an RS-232 interface the voltage has to be reduced, and the *low* and *high* voltage level inverted. In the other direction (sending data from some logic over RS-232) the low logic voltage has to be "bumped up", and a negative voltage has to be generated, too.

2.4.4 MAX 232 IC

The MAX232 was the first IC which in one package contains the necessary drivers (two) and receivers (also two), to adapt the RS-232 signal voltage levels to TTL logic. It became popular, because it just needs one voltage (+5V) and generates the necessary RS-232 voltage levels (approx. -10V and +10V) internally. This greatly simplified the design of circuitry. Circuitry designers no longer need to design and build a power supply with three voltages (e.g. -12V, +5V, and +12V), but could just provide one +5V power supply, e.g. with the help of a simple 78x05 voltage converter.

The circuitry is completed by connecting five capacitors to the IC as it follows. The MAX232 needs 1.0µF capacitors, the MAX232A needs 0.1µF capacitors. MAX232 clones show similar differences. It is recommended to consult the corresponding data sheet. At least 16V capacitor types should be used. If electrolytic capacitors are used, the polarity has to be observed.

2.4.5 SERIAL PORT

National Semiconductor has made the UART chips which have driven the PC's serial port ever since the emergence of IBM's first PC.

PC could receive and transmit data at speeds of up to 56 Kbit/s and, in the days of 4.77MHz bus speeds and serial printers, was perfectly adequate. When the IBM-AT came along a new UART was required because of the increase in bus speed and the fact that the bus was now 16 bits wide. All serial data transfers are handled by the CPU and each byte must pass through the CPU registers to get to memory or disk. This means that access times must be fast enough to avoid read overrun errors and transmission latency at higher bit rates. The speed at which data was routinely transmitted through the serial port was significantly less than is possible with modern modems.

The delay is the bus latency time associated with servicing the UART interrupt request. If the CPU cannot service the UART before the next data byte is received (by the UART from the serial port), data will be lost, with consequent retransmissions and an inevitable impact on throughput.

This condition is known as overrun error. At low bit rates the AT system is fast enough to read each byte from the UART receiver before the next byte is received. The higher the bit rate at the serial port, the higher the strain on the system to transfer each byte from the UART before the next is received. Higher bit rates causes the CPU to spend increasing amounts of time servicing the UART, thus making the whole system run inefficiently.

The previous problems by including First In First Out (FIFO) buffers on the receiver and transmitter, which dramatically improve performance on modem transfer speeds of 9.6 Kbit/s or higher.
The size of the receiver FIFO ensures that as many as 16 bytes are ready to transfer when the CPU services the UART receiver interrupt. The receiver can request transfer at FIFO thresholds of one, four, eight, 16 bytes full. This allows software to modify the FIFO threshold according to its current task and ensures that the CPU doesn't continually waste time switching context for only a couple of bytes of data received.

The transmitter FIFO ensures that as many as 16 bytes can be transferred when the CPU services the UART transmit interrupt. This reduces the time lost by the CPU in context switching. However, since a time lag in servicing an asynchronous transmitter usually has no penalty, CPU latency is of no concern when transmitting, although ultimate throughput may suffer.

PROJECT DESCRIPTION

3. PROJECT DESCRIPTION

3.1 Project Specification

The Objective of our Project is to watermark the Secondary Image with primary image for Security Purpose.

The Major Modules of the project are

1. Hardware Module
2. Software Module

I. Hardware Module:

The Inner Modules are

1. Power Supply
2. Serial Interface
3. Micro Controller

1. Power supply consists of a Transformer (Step Down) to transform the 230V AC to 5V AC. The Diode Bridge is used to convert AC to DC. Capacitors are used to reduce the noise and distortion. Voltage Regulator regulates the voltage and gives constant output voltage.

2. Serial Interface is used to convert the TTL to RS232 Logic which is understandable by system. It interfaces the data by VB.

3. Micro Controller consists of Assembly language code which gives the key to the software. We use 10th and 11th pin, which are used to receive and transmit the data respectively.

II. Software Module

The Inner Modules are

1. Java Editor
2. Trail Version
3. Full Version

1. We create a Java Editor by using Layouts. It consists of Buttons, Which is used to do the Specified Operation.

2. Trail Version is a version which consists of less number of features. We can see only the work of the project. There is another picture which is embedded with the watermarked image called Demo Version. We can not save the picture in this version.

3. Full Version is accessed by the hardware key. In this version, we save the picture in different file format. The key send by hardware matches with another file, gets full version of the project. The Demo picture is not embedded in this version.

3.2 IMPLEMENTATION

Implementation is one of the most important task in project development phase, in which one has to be cautious because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the Digital Image Watermarking with Hardware Key should be modified as a result of a programming. The system developed is completely menu driven. Further a simple operating procedure is provided so that the user can understand the different functions clearly and quickly.

The implementation of the project is done through following steps

a) Installing the windows XP with user permission
b) Establish VB in windows XP. Access the Full Version by invoking the Content of the Hardware manipulation using Components tool.

While we connecting the hardware key to the power supply and PC then the 230V power supply is given. The step down transformer reduces 230V AC to 5V AC. Bridge rectifier converts AC to DC. Voltage regulator is used to regulate the voltage and gives a constant output as 5V.Capacitors used in this circuit, reduce the noise. Then the 5V is goes to Microcontroller circuit. The Microcontroller sends the data to the serial Interface circuit. Serial Interface circuit needs 12V power supply. The capacitors acts as voltage pumpers and increase the voltage to 12V. Serial Interface circuit converts the data from TTL to RS 232 logic

Layouts are used to create a Java editor for the project. We use Border and Grid Layout. We browse the primary and secondary images to be watermarked. We separate RGB colors from each pixel of the pictures. We also calculate the weight, width and height of each color in RGB. We merge the pixel of primary and secondary image by using a formula.

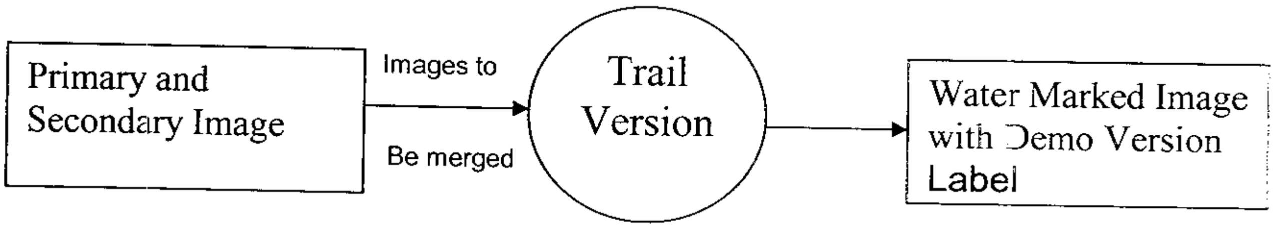
If it is trail version, then demo will also merged with the picture. Demo version doesn't have saving option. If the hardware key is connected then the project converts to full version. The code checks whether the hardware key is connected or not. Hardware key sends a data to system. It is interfaced by VB and stored as a text file in the system. The code checks the key with the first line of the text file.If both the text is equal then it converts to full version.

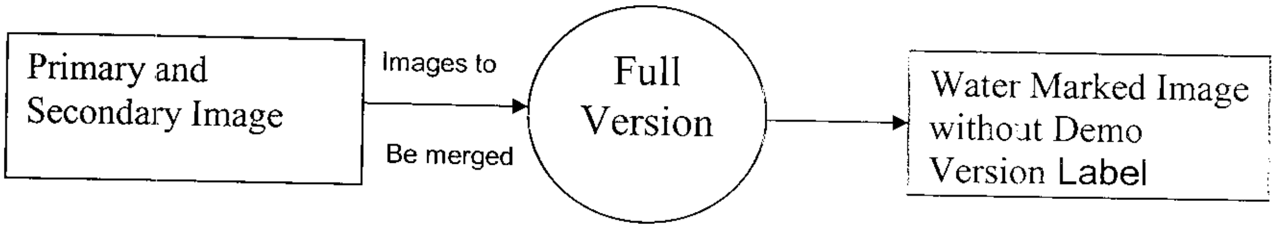
We add the COM component in the VB. We select the port of serial communication in the property page of COM object. The data come from hardware key is stored as text in the text box. A file is created in a particular path and the key is stored in that file. If the hardware key removed then the code will automatically the file will be deleted.

3.3 Logical Design

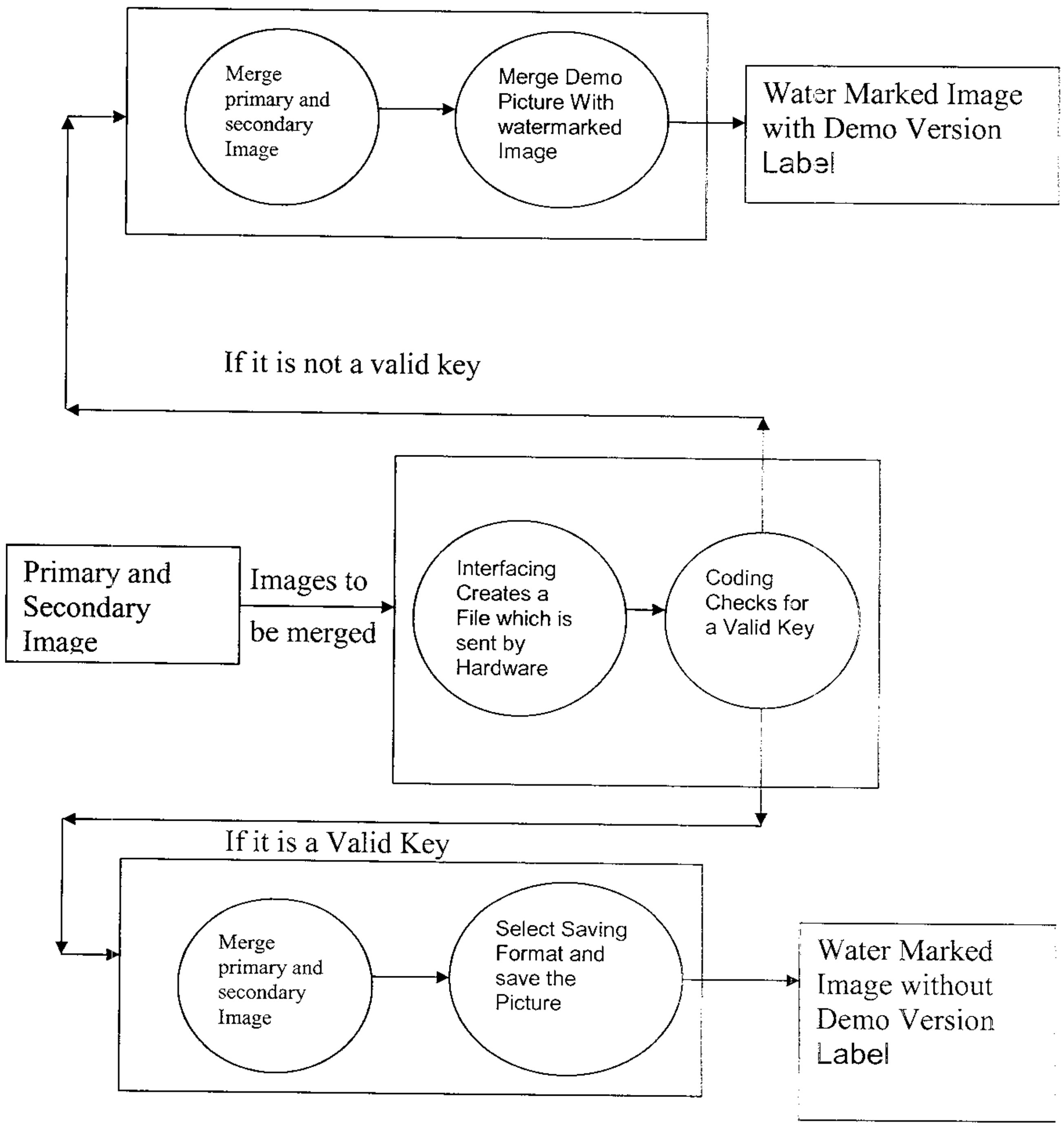
3.3.1 Data Flow Diagram

3.3.1.1 LEVEL 0


Primary and Secondary Image
Images to
Be merged
Trail Version
Water Marked Image with Demo Version Label

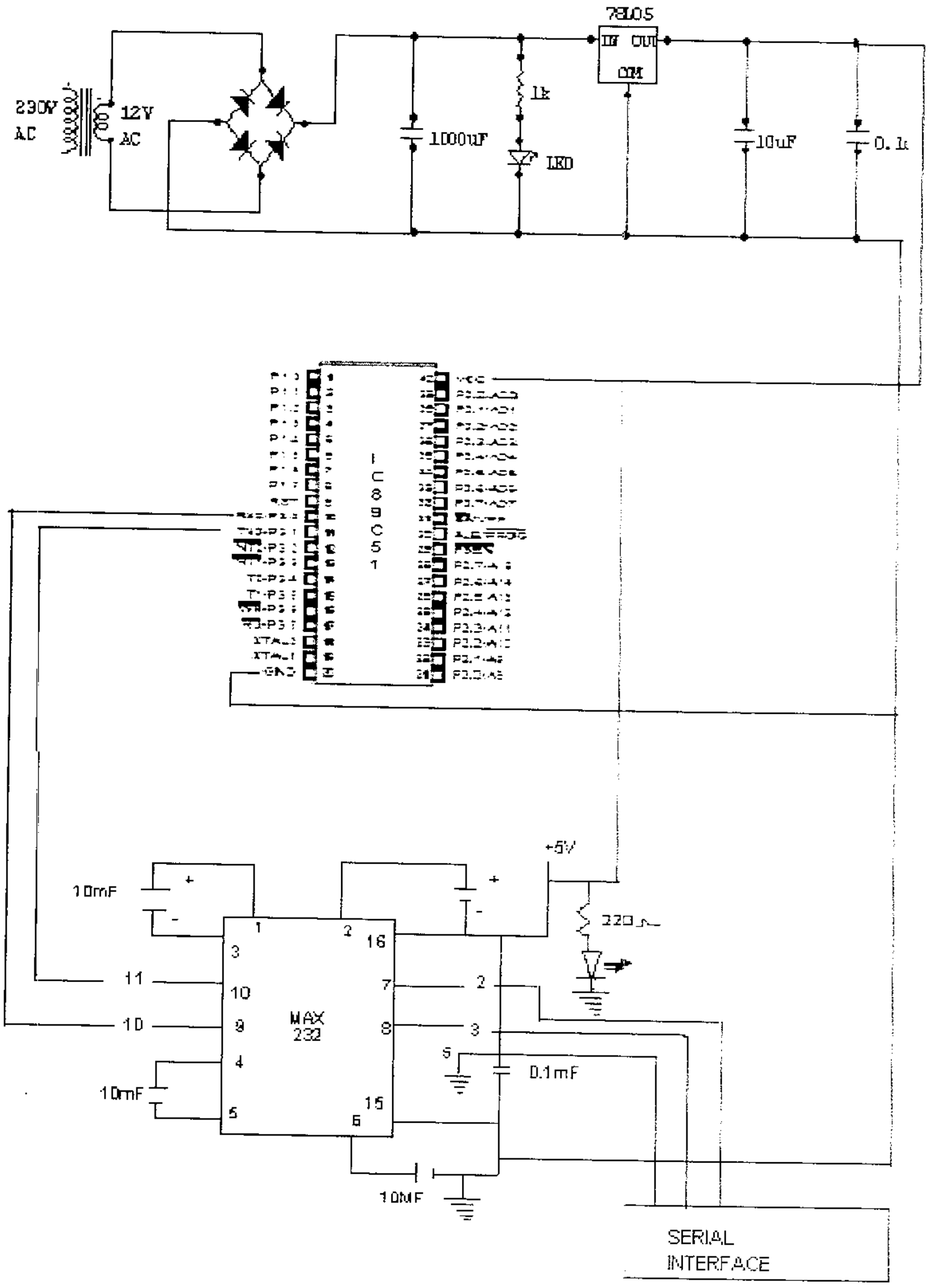

Primary and Secondary Image
Images to
Be merged
Full Version
Water Marked Image without Demo Version Label

3.3.1.2 LEVEL 1



Merge primary and secondary Image
Merge Demo Picture With watermarked Image
Water Marked Image with Demo Version Label
If it is not a valid key
Primary and Secondary Image
Images to be merged
Interfacing Creates a File which is sent by Hardware
Coding Checks for a Valid Key
If it is a Valid Key
Merge primary and secondary Image
Select Saving Format and save the Picture
Water Marked Image without Demo Version Label

3.3.2 CIRCUIT DIAGRAM



3.4 SYSTEM DESIGN

3.4.1 INPUT DESIGN

Input design is the process of converting user oriented inputs to a computer based format. The quality of the system input determines the quality of system output. Input design determines the format and validation criteria for data entering to the system.

Input design is a part of the overall system design, which requires very careful attention. If the data going into the system is incorrect then the processing and output will magnify these errors. Input can be categorized as internal, external, operational, computerized and interactive. The analysis phase should consider the impact of the inputs on the system as a whole and on the other systems.

The main objectives considered during input design are:

1. Nature of input processing
2. Flexibility and thoroughness of validation rules
3. Handling of priorities within the input document
4. Screen design to ensure accuracy and efficiency of input Relationship
   with files.
5. Careful design of the input also involves attention to error handling,
   Controls, batching and validation procedures.

The key created by the hardware are checked by the program and if it is matched then it is converted to full version. Any abnormality found in the inputs are checked and handled effectively. Input design features can ensure the reliability of a system and produce results from accurate data or they can result in the production of erroneous information. Messages and prompts should be well defined in this Input design. Labels for menus, items and fields should be clear.

3.4.2 OUTPUT DESIGN

Output generally refers to the results and information that are generated by the system. For many end users, outputs are the main reason for developing the system and the basis on which they will evaluate the usefulness of the application. Most end-users will noticeably operate the information system or enter data through workstation, but they will use the output from the system.

The output design determines what information should be present. It is also determines where to display, selection of output medium and to print the information.

Interfacing is done by using Visual basic component. Interfacing is taken place between Hardware key and coding. If the data send by hardware key matches with the key used in coding, then this project works as full version. Otherwise it works as a trail version. The Hardware key determines the output design.

COM gives an interface among the hardware key and the coding. Visual basic determines the output design of this project. If the key matches then the output design is full version. Otherwise it is trail version.

LITERATURE REVIEW

4. LITERATURE REVIEW

4.1 COMPONENTS

Component models are mainly of two types

1. **Visual Components**
2. **Non-visual components.**

Both can encapsulate either technical or business knowledge. The differences between the two are dependent on functionality. If the component provides only a benefit to the developer then the component is categorized as technical.

Ex: - a TCP/IP communication library.

Business components provide a benefit to the developer and end-user by encapsulating business knowledge.

Ex: - Address formatting and credit card validation components.

4.1.1 Visual Components

- ➢ In this project, it is able to create a complete User Interface on Layout or with in a Windows application.

4.1.2 NON VISUAL COMPONENTS

- ➢ It do not provide Pre-designed presentation interface to the user.
- ➢ In this, Interface is not visible through property page.

MERITS OF THE PROPOSED SYSTEM

5. MERITS OF THE PROPOSED SYSTEM:

5.1 ADVANTAGES:

- **Perceptual transparency**

  Watermarked content has the same subjective quality as the original contents.
- **Robustness**

  Nobody is able to remove, alter, or damage the Watermarked image.
- **Security**

  Hardware key enables the security for the software.

5.2 ENHANCEMENTS:

- Multiple Water Marking in a same time.
- We implement the watermark in the picture as we want.
- In the Trail Version, We only see the watermarked image. We cannot save it.
- Only in full version, we export the file in various extensions.

CONCLUSION

6. CONCLUSION

The project Digital Watermarking is designed and developed for secure the picture and software. The application is highly user friendly and the modules deal with the entire system processing. The application is developed using Java as Programming language with Visual Basic for interfacing and assembly language for microcontroller programming.

The project Digital Image Watermarking using Hardware key embeds copyright information such as author/owner/usage restrictions into the original file, be it an original photograph and it is a visible signature embedded inside an image to show authenticity or proof of ownership. It is not possible to remove, alter or damage the watermarked image. The purpose of digital image watermarking is to provide copyright protection for the images to be used on the web pages in the internet.

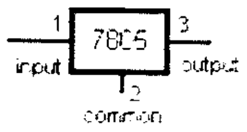
The system is very interactive so that the user can do the operation of creating watermarked picture very easily and effectively. Once an image is posted on a web page, the image is unprotected and can easily be copied and redistributed as an "original" if the image is watermarked then image used will be protected. The customer secures his pictures among the internet hackers. The code written in this project is very clear. The system is tested with various sample data. This package developed is tested with sample data, which provided the satisfactory results. After the system has been implemented, the maintenance of the system should be very easy so that the forthcoming changes can be made easily.

APPENDIX

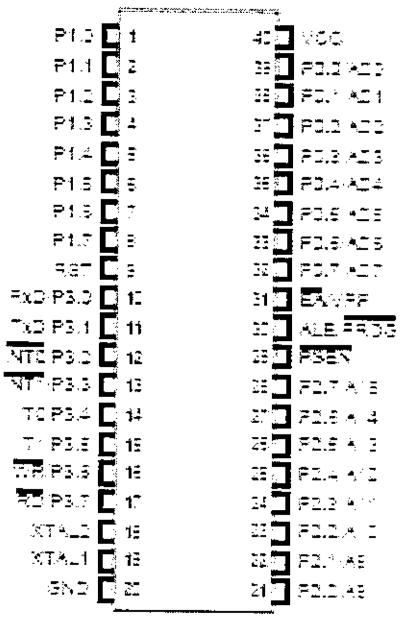
7. APPENDIX

a. PIN DIAGRAMS

i. LM 7805



ii. IC89C51



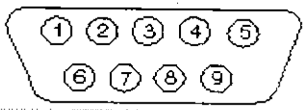
iii. MAX 232

```
          +---v---+
  C1+   -|1    16|-  Vcc
   V+   -|        |-  GND
  C1-   -|        |-  T1out
  C2+   -|        |-  R1in
  C2-   -|        |-  R1out
   V-   -|        |-  T1in
T2out   -|        |-  T2in
 R2in   -|8      9|-  R2out
          +-------+
```

IV. SERIAL CONNECTOR

9 Pin Connector on a DTE device (PC connection)

Male RS232 DB9



Pin Number	**Direction of signal:**
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request To Send (RTS) Outgoing flow control signal
8	Clear To Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a mode

SAMPLE CODING:

Visual Basic:

```
Dim a As Long
Dim fso As New FileSystemObject, fl As File

Private Sub Form_Load()
 MSComm1.PortOpen = True
End Sub

Private Sub Form_Unload(Cancel As Integer)
 MSComm1.PortOpen = False
End Sub

Private Sub MSComm1_OnComm()
 Text1.Text = MSComm1.Input

End Sub

Private Sub Timer1_Timer()
a = Val(Text1.Text)

On Error Resume Next
 If a = 12100145 Then
Set fl = fso.CreateTextFile("c:\watermark\FileName.txt",
True)
  Open "c:\watermark\FileName.txt" For Append As 1
  Write #1, a
  Close #1
Else
 Kill "c:\watermark\FileName.txt"
End If
 Text1.Text = ""
End Sub
```

Assembly Language:

```
baudnum         equ 0f3h

                org 0000h

                mov p3,#0ffh

                mov a,#00h

                anl pcon,#7fh
                        ;set SMOD bit to 0 for Buad*32 rate
                anl tmod,#30h
                        ;alter timer T1 configuration only
                orl tmod,#20h
                        ;set timer T1 as an 8-bit autoload

                mov th1,#baudnum
                        ;TH1 set for divide clock by 13d

                setb tr1
                        ;T1 running at 1E6/13=76923Hz
                mov scon,#40h

                setb ren

l1

        mov a,#"1"
        acall xmit

        mov a,#"2"
        acall xmit

        mov a,#"1"
        acall xmit

        mov a,#"0"
        acall xmit

        mov a,#"0"
        acall xmit
```

```
            mov a,#"1"
            acall xmit

            mov a,#"4"
            acall xmit

            mov a,#"5"
            acall xmit

            acall delay

            sjmp l1



xmit                  mov sbuf,a
xmit1
                      jbc ti,return
                      nop
                      sjmp xmit1

return                        ret


delay       mov r0,#01h
in2                   mov r1,#00h
in1                   mov r2,#00h
dwait                 djnz r2,dwait
                      djnz r1,in1
                      djnz r0,in2
                      ret
```

Java Code:

```
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import java.util.Enumeration;


class  Merge implements
ActionListener,WindowListener,ItemListener
{

Frame fr;
String Soption="";
String Aoption="Center";
Button BMerge,BQuit,BPBrowse,BPBrowse1;
Label
LMerge,LMerge1,LMerge2,LMerge3,LEmpty1,LEmpty2,LE
mpty3,Lab;
TextField TMerge,TMerge1;
String Fname="";
Label LogoLab,saveText;

Label LEmp,LEmp1,LEmp2,LEmp3,LEmp4;


TextField TLine,Topt;


  public Merge()
    {

        fr = new Frame("Water Marking ");

            BMerge =  new Button("Merge");
            BQuit = new Button("QUIT");
        BPBrowse = new Button("Select");
              BPBrowse1 = new Button("Select");


         TMerge = new TextField("",10);
              TMerge1 = new TextField("",10);


      TLine = new TextField("",20);
              Topt = new TextField("",20);


              BMerge.addActionListener(this);
              BQuit.addActionListener(this);

              BPBrowse.addActionListener(this);
              BPBrowse1.addActionListener(this);

              fr.addWindowListener(this);

              BMerge.setActionCommand("Merge");
              BQuit.setActionCommand("QUIT");

              BPBrowse.setActionCommand("PBROWSE");
              BPBrowse1.setActionCommand("PBROWSE1");



        Lab = new Label("Water Marking ");
             LogoLab = new Label("Logo Alignment ");
             saveText = new Label("Select Save Option ");

             Choice SeAlign = new Choice();
              SeAlign.addItem("Center");
              SeAlign.addItem("Left");
              SeAlign.addItem("Right");


SeAlign.addItemListener(this);

             LMerge = new Label("Select First Image to Water
Mark  : ");
             LMerge1 = new Label("Select Second Image to
Water Mark  : ");
```

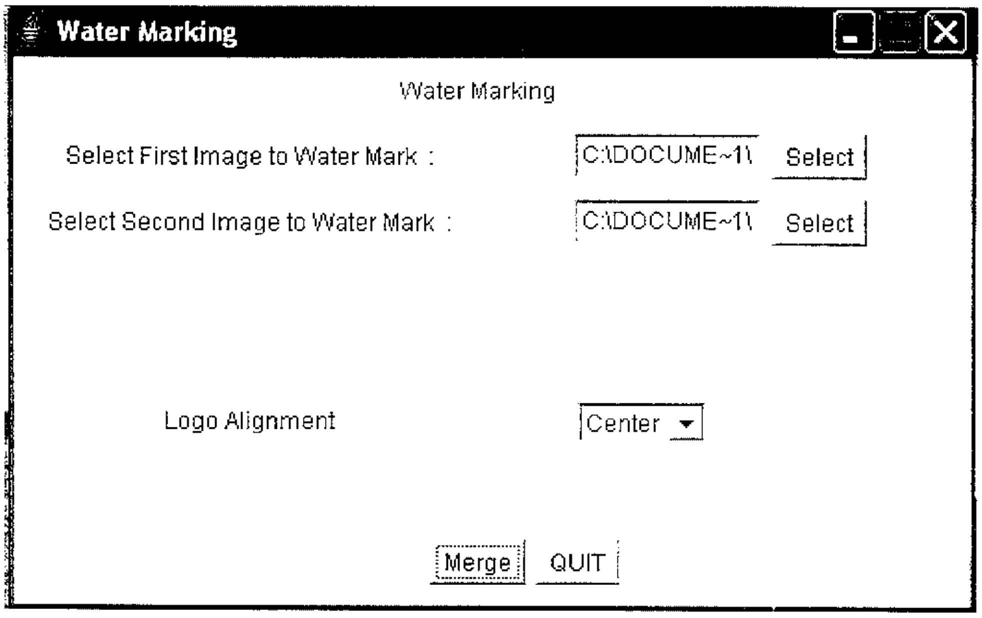
```
 LEmp = new Label("                    ");
        LEmp2 = new Label("          ");
        LEmp1 = new Label("                    ");
        LEmp3 = new Label("                    ");
        LEmp4 = new Label("                ");


 Panel pn = new Panel();

 fr.add("North",pn);
        pn.add(LEmp3);
        pn.add(Lab);
        pn.add(LEmp4);
 Panel p1 = new Panel();
 Panel p2 = new Panel();
 Panel p3 = new Panel();
 Panel p4 = new Panel();
        Panel p5 = new Panel();
        Panel p6 = new Panel();
        Panel p7 = new Panel();
        Panel p8 = new Panel();
        Panel p9 = new Panel();
        Panel p10 = new Panel();
        Panel p11 = new Panel();
        Panel p12 = new Panel();
        Panel p13 = new Panel();
        Panel p14 = new Panel();
        Panel p15 = new Panel();

 fr.add("Center",p1);

        p1.setLayout(new GridLayout(6,1));
 p1.add(p2);
 p1.add(p3);
 p1.add(p4);
 p1.add(p5);
        p1.add(p8);
 p1.add(p9);
        p1.add(p10);
 p1.add(p11);
        p1.add(p12);
        p1.add(p13);
```

```
                p1.add(p6);

        p2.add(LMerge);
                p3.add(TMerge);
                p3.add(BPBrowse);

        p4.add(LMerge1);
                p5.add(TMerge1);
                p5.add(BPBrowse1);


        p12.add(LogoLab);
                p13.add(SeAlign);
                p13.add(LEmp4);

fr.add("South",p6);
                p6.add(p7);
                p7.add(LEmp3);
                p7.add(BMerge);
                p7.add(BQuit);
                p7.add(LEmp2);

fr.add("East",p14);
fr.add("West",p15);


      fr.setResizable(false);
    fr.setLocation(320,30);
            fr.pack();
            fr.setVisible(true);

  }

public void actionPerformed(ActionEvent event)
    {

        String txt = event.getActionCommand();

            if (txt.equals("Merge"))
            {


                    String str = TMerge.getText();
                    String str1 = TMerge1.getText();
```

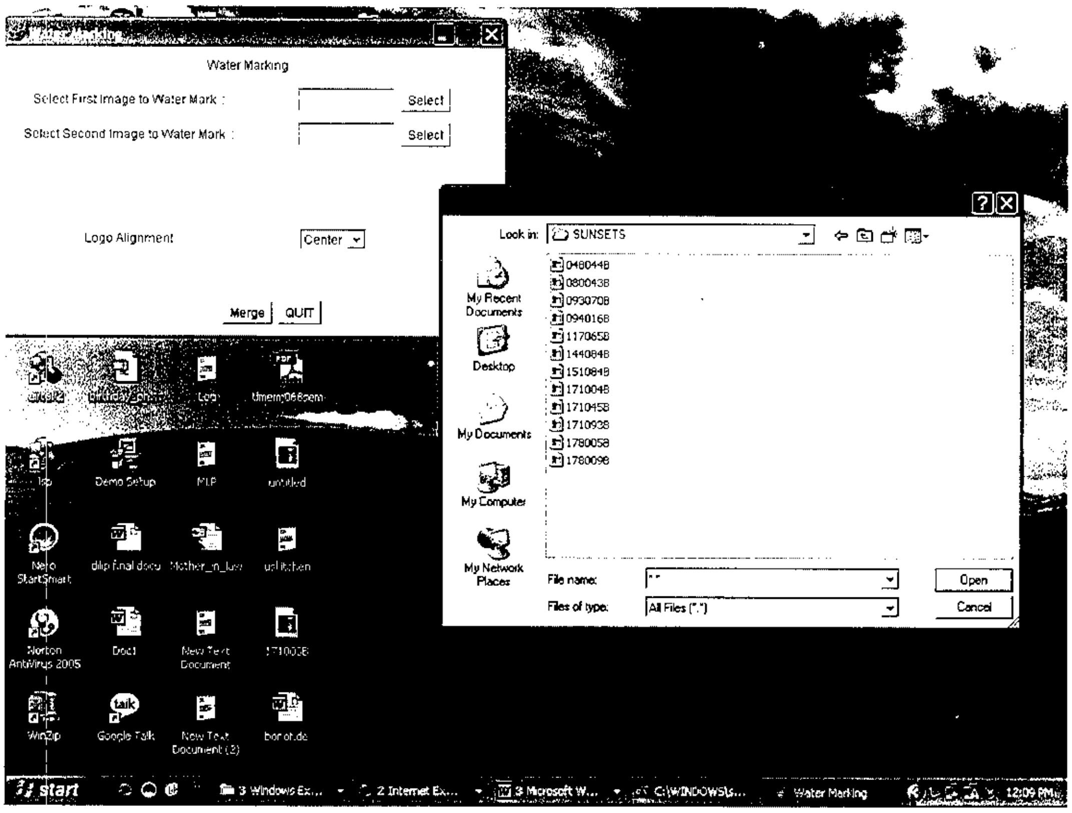
7 — 56

```
    String str2 ="";

                        double w1 = 0.5;
      double w2 = 0.5;


                     MergeImg icom = new MergeImg(str, w1,
 str1, w1,Aoption,str2);

              }


              if (txt.equals("PBROWSE"))
              {

      String ofile =  loadFile(fr, "BROWSE FILE TO
Merge",".\\", "*.*");


       TMerge.setText(ofile);
        }


     if (txt.equals("PBROWSE1"))
              {

      String ofile =  loadFile(fr, "BROWSE FILE TO
Merge",".\\", "*.*");


      TMerge1.setText(ofile);
       }


    if (txt.equals("QUIT"))
              {
           fr.setVisible(false);
fr.dispose();
  System.exit(0);

        }

    }
    public String loadFile(Frame f, String title,String defDir,
String fileType)
            {
  FileDialog fd = new FileDialog(f);
   fd.setFile(fileType);

   fd.setLocation(50, 50);
   fd.setVisible(true);
   String sFile = fd.getFile();

   File fi = new File(sFile);
     String s = fi.getPath();

   return fd.getDirectory() +
System.getProperty("file.separator").charAt(0) + fd.getFile();
   }

public static void main(String[] args)
     {
     Merge Open= new Merge();
   }

public void windowClosing(WindowEvent e) {
   fr.setVisible(false);
   fr.dispose();
   System.exit(0);
 }


public void windowClosed(WindowEvent e)
    {
     fr.setVisible(false);
   fr.dispose();
           System.exit(0);
    }

public void windowDeactivated(WindowEvent e) {}
 public void windowActivated(WindowEvent e) {}
 public void windowDeiconified(WindowEvent e) {}
 public void windowIconified(WindowEvent e) {}
 public void windowOpened(WindowEvent e) {}
public void itemStateChanged(ItemEvent event)
{
        Choice choice = (Choice)event.getSource();

        String selection = choice.getSelectedItem();

        String Aopt="";
        String Sopt="";

         if (selection.equals("Right")) {
                Aopt = "Right";
                }

         if (selection.equals("Left")) {
                Aopt = "Left";
                }

                if (selection.equals("Center")) {
                Aopt = "Center";
                }
        Aoption = Aopt;


}

}
```

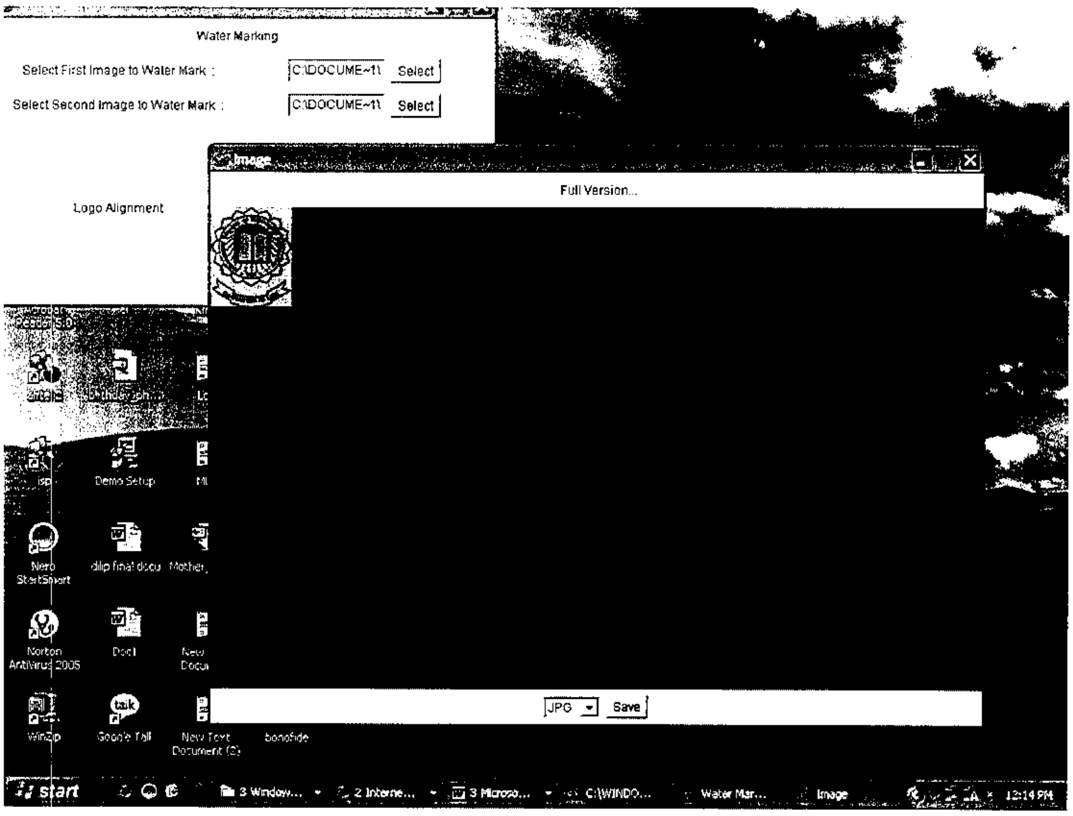
SCREENS:



Water Marking
Water Marking
Select First Image to Water Mark :
C:\DOCUME~1\
Select
Select Second Image to Water Mark :
C:\DOCUME~1\
Select
Logo Alignment
Center
Merge
QUIT

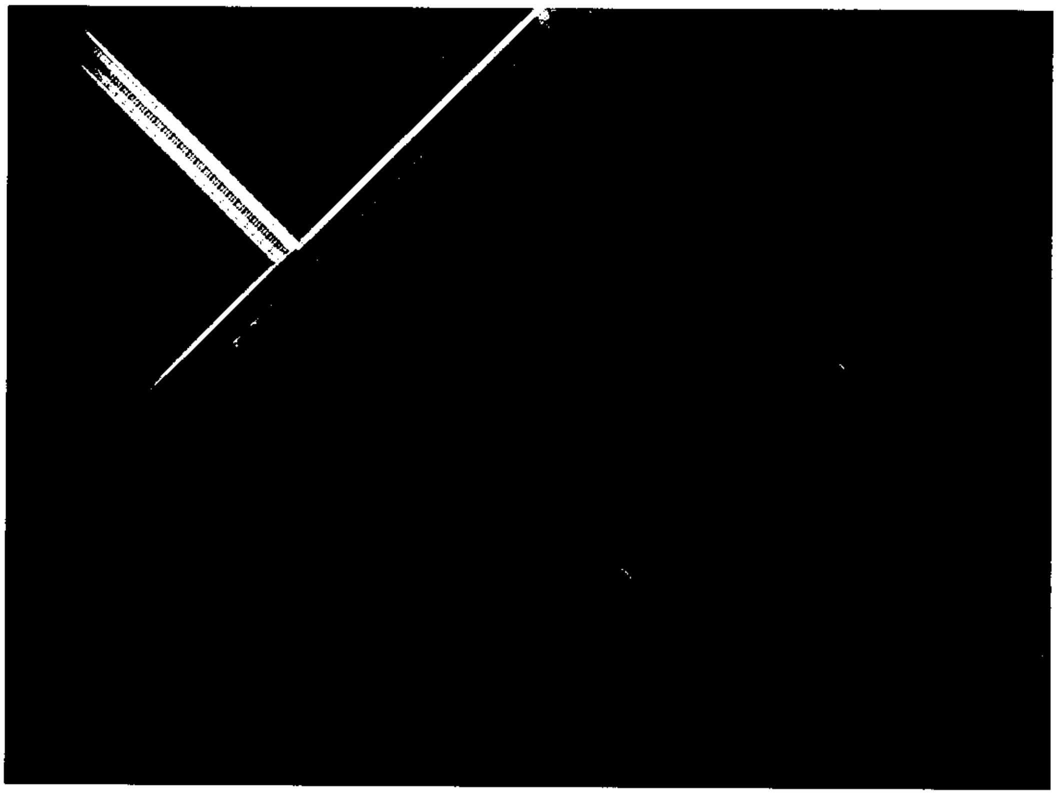


Water Marking
Select First Image to Water Mark :
Select
Select Second Image to Water Mark :
Select
Logo Alignment
Center
Merge
QUIT
Look in: SUNSETS
0480448
0800438
0930708
0940168
1170658
1440848
1510849
1710048
1710458
1710938
1780058
1780098
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
File name:
Files of type: All Files (*.*)
Open
Cancel
start
Water Marking
12:09 PM


Water Marking
Select First Image to Water Mark :
C:\DOCUME~1\
Select
Select Second Image to Water Mark :
C:\DOCUME~1\
Select
Logo Alignment
Image
Full Version...
JPG
Save
start
Water Mar...
Image
12:14 PM

PRIMARY IMAGE:

SECONARY IMAGE:



REFERENCES

8. REFERENCES

1. Gary Cornell, "Visual Basic 6 From the ground Up", TATA MCGRAW-HILL Publication (HILL Edition, 1999)
2. Dietel & Dietel," Java 2 How to Program", PEARSON Publication (Low Price Edition, 1999)
3. KennethJ.Ayala, "8051MicrocontrollerArchitect, Programming,& Application" PENRAM Publication (Second edition 1997)

WEBSITES:

1. http://www.microcontrollers.com	Referred Date: 9/9/2005
2. http://www.cosy.sbg.ac.at	Referred Date: 5/1/2006
3. http://www.research.ibm.com	Referred Date: 9/9/2005
4. http://www.beyondlogic.org	Referred Date: 25/1/2006
5. http://en.wikipedia.org	Referred Date: 5/3/2006