MAGNETIC STRIP AND BARCODE DECODING

Instructor:
Dr. Kwok-Bun Yue

Mentor:
Bruce Brenner

Team Members:
Islam, Rizwana
Mohammed, Mushtaq A.
Sheikh, Mudassir M.

Capstone Project Spring 2007
May 1, 2007
Abstract

MiniCheck ID aims to minimize the losses due to fake IDs such as the Drivers Licenses and State IDs. The application empowers a retail outlet’s sales representative by providing features for ID verification and validation.

In order to develop a solution that would be usable nation wide the recommendations proposed by the American Association of Motor Vehicle Administrators for magnetic stripe and barcode IDs are explored and based on these a flexible solution has been designed.

The application extracts the encoded information from magnetic stripe or barcode, parses the data and saves it to the database. In case of magnetic ID information is retrieved just by swiping the card on the reader whereas for the barcode cards image is scanned using standard card scanner and a third part software [Clear Image Software Development Kit] is used to convert that image to raw data.

For magnetic stripe cards a revolutionary technology called MagnePrint developed by Magtek Inc is used to validate the authenticity of a card. Magnetic stripe are constructed (naturally by the production process) of billions of particles of varying size and shape forming a unique pattern (these exceptional patterns are as distinct as a fingerprint); this technology captures these patterns. Magtek has manufactured special purpose readers that can interpret these patterns and convert them into MagnePrint signature. Our application uses MagTek’s web interface to access and validate the MagnePrint signatures. One area of potential enhancement is the validation of MagnePrint signature from within the application using API calls.

The application also provides a feature for printing the stored information in the database through a report. This report can be exported in various formats such as MS Excel, MS Word, etc.
# Table of Contents

1. Overview .................................................................................................................... 4

2. Features .................................................................................................................... 4

3. Assumptions and Dependencies ............................................................................... 5

4. Project Requirements ............................................................................................... 5

5. Design and Architecture ........................................................................................... 6

6. Database .................................................................................................................. 8

7. Software Development Model ................................................................................... 8

8. Evaluation and Testing ............................................................................................. 8

9. Conclusion ............................................................................................................... 11

10. References ............................................................................................................. 11

11. Appendices ............................................................................................................ 12
    A. Member Information .............................................................................................. 12
    B. Application Screen Shots .................................................................................... 12
    C. Work Breakdown ................................................................................................. 15
    D. Source Code ........................................................................................................ 16
1. Overview

MiniCheck-OCR developed a software product called MiniCheck MGR to scan, validate and process checks and barcode IDs. We have developed a new standalone product MiniCheck ID to compliment the functionality. MiniCheck MGR was restricted to selected scanners whereas our solution integrates any scanner that MS Windows can recognize; in addition to this now we have also catered magnetic IDs.

A magnetic strip card is a plastic / PVC card with a medium that can be used to save stream of alpha numeric information. This magnetic medium is very similar to that in audio and VHS tapes. This medium is prone to loose data if exposed to a magnet and this is the only major concern for this technology.

A magnetic strip reader is a piece of hardware that can read information stored on the magnetic strip of the card by sliding the card through the equipment. This reader can be interfaced to a computer via a USB (Universal Serial Bus) port and the data on the card can be transferred to a computer.

Once the data is in the computer the software can make all types of data manipulations.

2. Features

The emphasis of the product is to accomplish the following tasks.

- Verifying Magnetic and Barcode IDs: The information printed on the ID can be compared to the stored information in the magnetic strips or barcodes.
- Validating Magnetic Cards: The MagnePrint signature adds an extra level of protection against duplicated or altered IDs. The application interfaces with MagneSafe’s web interface to register and authenticate cards.
- Storing Customers Information: The customer’s information saved for focused / targeted marketing and or validation.
- Printing Information: The information stored in the database can be printed and exported to various formats such as MS Word, MS Excel, etc.

3. Assumptions and Dependencies

The software is developed for the Microsoft Windows XP platform. All testing is performed on Windows XP Professional with Service Pack 2.0 and .Net Framework 2.0. The ID images that are scanned for 2D barcode decoding must be scanned in 300dpi [most commercial card readers scan at 300dpi, selection of hardware is critical]

4. Project Requirements

The application allows the user to swipe and recognize a magnetic card, to scan and recognize barcode of ID cards or to use MagnetPrint to validate a magnetic card. It provides a modern user interface for selecting one of these actions. Both magnetic card and barcode information of a driver license is parsed using the standards defined by AAMVA [AM01]. The MagnePrint option allows the user to log on to the MageSafe website, scan a magnetic card and view the validation results.

The card reader is interfaced to the computer and the information is extracted off the magnetic strip. Software is developed to parse data into their respective fields.

The card holder’s information is displayed on the graphical user interface and stored to the database. Some extra pieces of information; such as the card holder’s date of birth is used to compute age. The MagnePrint[MG01] signature can be displayed on the interface and send for validation.
A final packaging and deployment application for MiniCheckID is created so that it can be distributed to clients incorporating all its dependencies and third party resources. The purpose of this is to minimize the steps required to get the application working and having the software bundled in a single package.

5. **Design and Architecture**

The system is developed using Microsoft C# [MS01]. The application is a standalone program that handles the Magnetic and 2D Barcode State IDs and Drivers Licenses. A user can select from the given four options [Magnetic Encoding, Barcode Encoding, Report generation or MagnePrint validation]. The first two options deal directly with decoding IDs, parse, store and display results. The third option presents the data stored in the form of a report. The fourth option can be used to validate a magnetic card using MagneSafe’s web interface.
The inputs of the system include raw data from the magnetic stripe reader or a scanned image of an ID. In case of an image the software uses a Clear Image Software Development Kit [CL01] to convert image to raw data. The SDK requires that the image be scanned in at least 300dpi and if the picture scanned is not straight the conversion process has to be performed again.

Once the raw data is available the string is parsed i.e. broken down into pieces of information and dumped into the local database.

In case of a magnetic card the MagnePrint[MG01] signature is forwarded to Magtek’s
Server for authentication.

6. Database

The system is designed for Microsoft Access [MS01] database. One table is used to store the card holder’s information. This can be considered as a flat file data repository without any relationships. The database being used is the same as MiniCheck MGR so the information being saved can be shared between the two applications.

7. Software Development Model

Water fall model has been selected for development as it is the ideal approach for the given time, complexity, available resources and team size for this project. The requirements and outcome are specified and documented clearly that favor the use of water fall model.

8. Evaluation and Testing

Seven samples of barcode IDs and a sample of magnetic ID are used as test cases. Ad-hoc manual testing has been performed and AAMVA’s specifications are strictly followed.
When a Texas driver license or a State ID [which is a magnetic card] is swiped on a regular three track reader the information received is as under:

%TXDALLAS^GATES$BRIAN$A^900 BAY AREA BLVD. #412^?;63601512345678=110918720911?#" 760580000  C A M811195BLABRN 2*)TI  

The data contains three tracks of information separated by question marks. Track 1 and 3 contain alpha-numeric information whereas track 2 only contains numeric information. The contents of each track are as under:

<table>
<thead>
<tr>
<th>Track 1</th>
<th>Track 2</th>
<th>Track 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>State and City</td>
<td>ISO Issuer Identification</td>
<td>Zip Code</td>
</tr>
<tr>
<td>Name (Last$First$Middle)</td>
<td>Number (IIN)</td>
<td>Class</td>
</tr>
<tr>
<td>Street Address</td>
<td>License Number</td>
<td>Restrictions</td>
</tr>
<tr>
<td></td>
<td>Expiry Date (yymm)</td>
<td>Gender</td>
</tr>
<tr>
<td></td>
<td>Date of Birth (yyyymmdd)</td>
<td>Height</td>
</tr>
</tbody>
</table>

Table 1: Magnetic ID [with 3 tracks]
If the same Texas ID is swiped on a MagnePrint enabled reader we get the following result:

| %TXDALLAS^GATES&BRIAN$A^900 BAY AREA BLVD. #412^;:00000000000000=11090000000?
| [0300]231E50402360038EF7321CA35412994F6305BF1624C9F6DF4BCBB942C2D4FC51BD0A6340973E1C148F3EE883B47D7DAC01A5564B428B1988F2F34EEA112F77|BE09F5260A5F89CA1C6E1FBF007551E2DC7F324A2CE3DC4BBD19F4C682DC0B8||002005A1|6E2CF1E3DF49CA59622F593D1241770FF82439B2D97342B4E8A40CBF600C446C08DBAB9309CEE10DAB21329D5BC035BEC124587A93AC8||6623C6C167A25872|4130344E3734320000048|E706|B6530E3BA6826717

Data contains two tracks and the MagnePrint signature. The special purpose card reader converts the pattern of magnetic stripe particles in a 54 byte MagnePrint signature.

<table>
<thead>
<tr>
<th>Track 1</th>
<th>Track 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>State and City</td>
<td>Expiry Date (yymm)</td>
</tr>
<tr>
<td>Name (Last$First$Middle)</td>
<td>MagnePrint Signature</td>
</tr>
<tr>
<td>Street Address</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Magnetic ID [with 2 tracks and MagnePrint]

In case of barcode IDs the image of the ID is scanned and is passed to the software development kit. The SDK returns the information in the form of the following string:

| @AAMVA63603301DL00270194DAQ999000680DAASAMPLE,SUSAN,DAGP.O. BOX 1272 ROOM 2120DAILITTLEROCKDAJARDAK72205DARDDASDATDAU511DAW121DAYBRDAZBRDBA20031212DBB19701212DBCFDBD20010416 |

The contents of the aforementioned string are as under:

<table>
<thead>
<tr>
<th>DAQ: License Number</th>
<th>DAK: Zip code</th>
<th>DAW: Weight</th>
<th>DBB: Date of Birth</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAR: Class</td>
<td>DAZ: Hair Color</td>
<td>DBC: Gender</td>
<td></td>
</tr>
<tr>
<td>DAS:</td>
<td>DAZ: Hair Color</td>
<td>DBD: Date of Issue</td>
<td></td>
</tr>
<tr>
<td>Restriction</td>
<td>DBA: Expiration Date</td>
<td>Issuer Identification</td>
<td></td>
</tr>
<tr>
<td>DAT:</td>
<td>DBK: Social Security Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Endorsement</td>
<td>DAU: Height</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAU: Height</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Barcode ID

These samples have determined the accuracy of the model. The parser extracts all the data items present in all cases where the AAMVA’s conventions are followed.
9. Conclusion

The project was an excellent experience for the team. One area that requires continued study is the MagnePrint [MG01] technology. The application can be extended to validate any type of magnetic cards included but not limited to financial cards. The team has stepped beyond the stated initial requirements of the project by incorporating barcode decoding and developing an application from scratch.

10. References

- [MR01] MiniCheck-OCR. http://www.minicheckocr.com/
11. Appendices

A. Member Information

The group members include:

Mushtaq A. Mohammed
   Email: mamushtaq01@gmail.com
   Major: Computer Information Systems
   Roles: Chief Analyst, DBA, Programmer

Rizwana Islam
   Email: deena76bd@yahoo.com
   Major: Computer Information Systems
   Roles: Web Master, Domain Expert, Programmer

Mudassir M. Sheikh
   Email: sheikhm7408@uhcl.edu
   Major: Computer Science
   Roles: Project Manager, Programmer

B. Application Screen Shots

![Figure 1: Magnetic Stripe Decoding using regular reader](image-url)
Figure 2: Magnetic Stripe Decoding using MagnePrint enabled reader

Figure 3: Barcode Decoding
Figure 4: MagnePrint signature verified using web interface

Figure 5: MagnePrint signature denied using web interface
C. Work Breakdown

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team Formation and Project Selection</td>
<td>1 day</td>
<td>01/18/07</td>
<td>01/18/07</td>
</tr>
<tr>
<td>Requirements and Specifications</td>
<td>10 days</td>
<td>01/18/07</td>
<td>01/31/07</td>
</tr>
<tr>
<td>Web Site Management and Project Documentation</td>
<td>71 days</td>
<td>01/19/07</td>
<td>04/26/07</td>
</tr>
<tr>
<td>Meetings and feedback from Instructor and Mentor</td>
<td>72 days</td>
<td>01/18/07</td>
<td>04/26/07</td>
</tr>
<tr>
<td>Technology Identification</td>
<td>17 days</td>
<td>02/02/07</td>
<td>02/24/07</td>
</tr>
<tr>
<td>Prototype Development</td>
<td>24 days</td>
<td>02/24/07</td>
<td>03/28/07</td>
</tr>
<tr>
<td>Deliver final product</td>
<td>16 days</td>
<td>03/28/07</td>
<td>04/18/07</td>
</tr>
</tbody>
</table>

Table 4: Work Breakdown

Figure 7: Gantt Chart representing major tasks
### Task Name

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Rizwana</th>
<th>Mushtaq</th>
<th>Mudassir</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements and Specifications</td>
<td>55.00%</td>
<td>25.00%</td>
<td>20.00%</td>
</tr>
<tr>
<td>Web Site Management and Project Documentation</td>
<td>70.00%</td>
<td>20.00%</td>
<td>10.00%</td>
</tr>
<tr>
<td>Technology Identification</td>
<td>11.66%</td>
<td>36.66%</td>
<td>51.68%</td>
</tr>
<tr>
<td>Prototype Development</td>
<td>20.00%</td>
<td>35.00%</td>
<td>45.00%</td>
</tr>
<tr>
<td>Deliver final product</td>
<td>10.00%</td>
<td>50.00%</td>
<td>40.00%</td>
</tr>
</tbody>
</table>

Table 5: Member Contributions

### D. Source Code

```csharp
// Libraries being used.
using System;
using System.IO;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;

namespace MSBC
{
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Timers.Timer timer1;
        private SplitContainer splitContainer1;
        private TabControl dl;
        private TabPage tabPage1;
        internal Button bm;
        internal TextBox tbm;
        private TextBox tbMagnetic;
        private TabPage tabPage2;
        internal PictureBox PictureBox1;
        internal TextBox txtRslt;
        internal Button cmdScan;
        private TabPage tabPage3;
        private WebBrowser wb;
        private Label label1;
        private Button bMagnetic;
        private Button bBarcode;
        private Button bMagnePrint;
        private Label label2;
        private Label label3;
    }
}
```

// Clear Image SDK's Library
using EOpType = nsCiFnc.OpStats.EOpType;

```csharp
namespace MSBC
{
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Timers.Timer timer1;
        private SplitContainer splitContainer1;
        private TabControl dl;
        private TabPage tabPage1;
        internal Button bm;
        internal TextBox tbm;
        private TextBox tbMagnetic;
        private TabPage tabPage2;
        internal PictureBoxPictureBox1;
        internal TextBox txtRslt;
        internal Button cmdScan;
        private TabPage tabPage3;
        private WebBrowser wb;
        private Label label1;
        private Button bMagnetic;
        private Button bBarcode;
        private Button bMagnePrint;
        private Label label2;
        private Label label3;
    }
}
```
private Label label4;
internal Button bb;
private Button bReport;
private TabPage tabPage4;
private Label label15;
private report report1;
private System.ComponentModel.Container components = null;

// Default Constructor
public Form1()
{
    InitializeComponent();
}

#region Windows Form Designer generated code

private void InitializeComponent()
{
    System.ComponentModel.ComponentResourceManager resources =
        new System.ComponentModel.ComponentResourceManager(typeof(Form1));
    this.OpenFileDialog1 = new System.Windows.Forms.OpenFileDialog();
    this.timer1 = new System.Timers.Timer();
    this.splitContainer1 = new System.Windows.Forms.SplitContainer();
    this.bMagnePrint = new System.Windows.Forms.Button();
    this.bBarcode = new System.Windows.Forms.Button();
    this.bMagnetic = new System.Windows.Forms.Button();
    this.label1 = new System.Windows.Forms.Label();
    this.dl = new System.Windows.Forms.TabControl();
    this.tabPage1 = new System.Windows.Forms.TabPage();
    this.label2 = new System.Windows.Forms.Label();
    this.bm = new System.Windows.Forms.Button();
    this.tbm = new System.Windows.Forms.TextBox();
    this.tbMagnetic = new System.Windows.Forms.TextBox();
    this.tabPage2 = new System.Windows.Forms.TabPage();
    this.bb = new System.Windows.Forms.Button();
    this.label3 = new System.Windows.Forms.Label();
    this.PictureBox1 = new System.Windows.Forms.PictureBox();
    this.txtRslt = new System.Windows.Forms.TextBox();
    this.cmdScan = new System.Windows.Forms.Button();
    this.tabPage3 = new System.Windows.Forms.TabPage();
    this.label4 = new System.Windows.Forms.Label();
    this.wb = new System.Windows.Forms.WebBrowser();
    this.tabPage4 = new System.Windows.Forms.TabPage();
    this.report1 = new MSBC.report();
    this.label5 = new System.Windows.Forms.Label();

    ((System.ComponentModel.ISupportInitialize)(this.timer1)).BeginInit();
    this.splitContainer1.Panel1.SuspendLayout();
this.splitContainer1.Panel2.SuspendLayout();
this.splitContainer1.SuspendLayout();
this.d1.SuspendLayout();
this tabPage1.SuspendLayout();
this tabPage2.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(this.PictureBox)
1).BeginInit();
this tabPage3.SuspendLayout();
this tabPage4.SuspendLayout();
this.SuspendLayout();

// timer1
//
this.timer1.AutoReset = false;
this.timer1.SynchronizingObject = this;
this.timer1.Elapsed += new
System.Timers.ElapsedEventHandler(this.timer1_Elapsed);

// System.EventHandler(this.bReport_Click);

// bMagnePrint
//
this.bMagnePrint.Location = new System.Drawing.Point(0,
139);
this.bMagnePrint.Name = "bMagnePrint";
this.bMagnePrint.Size = new System.Drawing.Size(93, 58);
this.bMagnePrint.TabIndex = 3;
this.bMagnePrint.Text = "MagnePrint";
this.bMagnePrint.UseVisualStyleBackColor = true;
this.bMagnePrint.Click += new
System.EventHandler(this.bMagnePrint_Click);

// bBarcode
//
this.bBarcode.Location = new System.Drawing.Point(0, 81);
this.bBarcode.Name = "bBarcode";
this.bBarcode.Size = new System.Drawing.Size(93, 58);
this.bBarcode.TabIndex = 2;
this.bBarcode.Text = "Barcode";
this.bBarcode.UseVisualStyleBackColor = true;
this.bBarcode.Click += new
System.EventHandler(this.bBarcode_Click);

// bMagnetic
//
this.bMagnetic.Location = new System.Drawing.Point(0, 23);
this.bMagnetic.Name = "bMagnetic";
this.bMagnetic.Size = new System.Drawing.Size(93, 58);
this.bMagnetic.TabIndex = 1;
this.bMagnetic.Text = "Magnetic";
this.bMagnetic.UseVisualStyleBackColor = true;
this.bMagnetic.Click += new
System.EventHandler(this.bMagnetic_Click);
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label1.Location = new System.Drawing.Point(0, 0);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(93, 23);
this.label1.TabIndex = 0;
this.label1.Text = "ID Decoding";
this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

this.dl.Controls.Add(this.tabPage1);
this.dl.Controls.Add(this.tabPage2);
this.dl.Controls.Add(this.tabPage3);
this.dl.Controls.Add(this.tabPage4);
this.dl.Location = new System.Drawing.Point(0, 0);
this.dl.Multiline = true;
this.dl.Name = "dl";
this.dl.SelectedIndex = 0;
this.dl.Size = new System.Drawing.Size(797, 491);
this.dl.TabIndex = 16;

this.tabPage1.Controls.Add(this.label2);
this.tabPage1.Controls.Add(this.bm);
this.tabPage1.Controls.Add(this.tbm);
this.tabPage1.Controls.Add(this.tbMagnetic);
this.tabPage1.Location = new System.Drawing.Point(4, 4);
this.tabPage1.Name = "tabPage1";
this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
this.tabPage1.Size = new System.Drawing.Size(789, 465);
this.tabPage1.TabIndex = 0;
this.tabPage1.UseVisualStyleBackColor = true;

this.tabPage1.Controls.Add(this.label2);
this.tabPage1.Controls.Add(this.bm);
this.tabPage1.Controls.Add(this.tbm);
this.tabPage1.Controls.Add(this.tbMagnetic);
this.tabPage1.Location = new System.Drawing.Point(4, 4);
this.tabPage1.Name = "tabPage1";
this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
this.tabPage1.Size = new System.Drawing.Size(789, 465);
this.tabPage1.TabIndex = 0;
this.tabPage1.UseVisualStyleBackColor = true;

System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label2.Location = new System.Drawing.Point(3, 3);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(783, 23);
this.label2.TabIndex = 21;
this.label2.Text = "Magnetic Decoding";
this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

this.bm.Enabled = false;
this.bm.Location = new System.Drawing.Point(9, 434);
this.bm.Name = "bm";
this.bm.Size = new System.Drawing.Size(765, 24);
this.bm.TabIndex = 20;
this.bm.Text = "Process / Parse";
this.bm.Click += new System.EventHandler(this.bm_Click);

System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.tbm.Location = new System.Drawing.Point(6, 107);
this.tbm.Multiline = true;
this.tbm.Name = "tbm";
this.tbm.ReadOnly = true;
this.tbm.Size = new System.Drawing.Size(768, 320);
this.tbm.TabIndex = 19;

this.tbMagnetic.Location = new System.Drawing.Point(6, 30);
this.tbMagnetic.Multiline = true;
this.tbMagnetic.Name = "tbMagnetic";
this.tbMagnetic.Size = new System.Drawing.Size(768, 71);
this.tbMagnetic.TabIndex = 0;
this.tbMagnetic.Enter += new System.EventHandler(this.tbMagnetic_Enter);

this.tabPage2.Controls.Add(this.bb);
this.tabPage2.Controls.Add(this.label3);
this.tabPage2.Controls.Add(this.PictureBox1);
this.tabPage2.Controls.Add(this.txtRslt);
this.tabPage2.Controls.Add(this.cmdScan);
this.tabPage2.Location = new System.Drawing.Point(4, 4);
this.tabPage2.Name = "tabPage2";
this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
this.tabPage2.Size = new System.Drawing.Size(789, 465);
this.tabPage2.TabIndex = 1;
this.tabPage2.UseVisualStyleBackColor = true;

this.tabPage2.Controls.Add(this.bb);
this.tabPage2.Controls.Add(this.label3);
this.tabPage2.Controls.Add(this.PictureBox1);
this.tabPage2.Controls.Add(this.txtRslt);
this.tabPage2.Controls.Add(this.cmdScan);
this.tabPage2.Location = new System.Drawing.Point(4, 4);
this.tabPage2.Name = "tabPage2";
this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
this.tabPage2.Size = new System.Drawing.Size(789, 465);
this.tabPage2.TabIndex = 1;
// bb
//
this.bb.Enabled = false;
this.bb.Location = new System.Drawing.Point(17, 432);
this.bb.Name = "bb";
this.bb.Size = new System.Drawing.Size(756, 25);
this.bb.TabIndex = 21;
this.bb.Text = "Process / Parse";
this.bb.Click += new System.EventHandler(this.bb_Click);
//
// label3
//
this.label3.Location = new System.Drawing.Point(3, 3);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(783, 23);
this.label3.TabIndex = 20;
this.label3.Text = "Barcode Decoding";
this.label3.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
//
// PictureBox1
//
this.PictureBox1.Location = new System.Drawing.Point(566, 65);
this.PictureBox1.Name = "PictureBox1";
this.PictureBox1.Size = new System.Drawing.Size(207, 356);
this.PictureBox1.TabIndex = 19;
this.PictureBox1.TabStop = false;
//
// txtRslt
//
this.txtRslt.Location = new System.Drawing.Point(17, 65);
this.txtRslt.Multiline = true;
this.txtRslt.Name = "txtRslt";
this.txtRslt.ReadOnly = true;
this.txtRslt.Size = new System.Drawing.Size(536, 356);
this.txtRslt.TabIndex = 18;
//
// cmdScan
//
this.cmdScan.Location = new System.Drawing.Point(17, 34);
this.cmdScan.Name = "cmdScan";
this.cmdScan.Size = new System.Drawing.Size(756, 25);
this.cmdScan.TabIndex = 16;
this.cmdScan.Text = "Scan";
this.cmdScan.Click += new System.EventHandler(this.cmdScan_Click);

// tabPage3
this.tabPage3.Controls.Add(this.label4);
this.tabPage3.Controls.Add(this.wb);
this.tabPage3.Location = new System.Drawing.Point(4, 4);
this.tabPage3.Name = "tabPage3";
this.tabPage3.Padding = new System.Windows.Forms.Padding(3);
this.tabPage3.Size = new System.Drawing.Size(789, 465);
this.tabPage3.TabIndex = 2;
this.tabPage3.UseVisualStyleBackColor = true;

// label4
this.label4.Location = new System.Drawing.Point(3, 3);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(783, 23);
this.label4.TabIndex = 16;
this.label4.Text = "MagnePrint Validation";
this.label4.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

// wb
this.wb.Location = new System.Drawing.Point(3, 29);
this.wb.MinimumSize = new System.Drawing.Size(20, 20);
this.wb.Name = "wb";
this.wb.Size = new System.Drawing.Size(780, 431);
this.wb.TabIndex = 15;
this.wb.Url = new System.Uri("https://access.magneprint.com/mpdukptkb/", System.UriKind.Absolute);
this.wb.WebBrowserShortcutsEnabled = false;

// tabPage4
this.tabPage4.Controls.Add(this.crv);
this.tabPage4.Controls.Add(this.label5);
this.tabPage4.Location = new System.Drawing.Point(4, 4);
this.tabPage4.Name = "tabPage4";
this.tabPage4Padding = new System.Windows.Forms.Padding(3);
this.tabPage4.Size = new System.Drawing.Size(789, 465);
this.tabPage4.TabIndex = 3;
this.tabPage4.UseVisualStyleBackColor = true;

// crv
this.crv.ActiveViewIndex = 0;
this.crv.DisplayGroupTree = false;
this.crv.Location = new System.Drawing.Point(3, 26);
this.crv.Name = "crv";
this.crv.ReportSource = this.report1;
this.crv.ShowCloseButton = false;
this.crv.ShowGotoPageButton = false;
this.crv.ShowGroupTreeButton = false;
this.crv.Size = new System.Drawing.Size(783, 436);
this.crv.TabIndex = 18;

System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label5.Location = new System.Drawing.Point(3, 3);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(783, 23);
this.label5.TabIndex = 17;
this.label5.Text = "Report";
this.label5.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(898, 472);
this.Controls.Add(this.splitContainer1);
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
this.Icon = ((System.Drawing.Icon)(resources.GetObject("this.Icon")));
this.MaximizeBox = false;
this.Name = "Form1";
this.Text = "MiniCheck ID";
this.Load += new System.EventHandler(this.Form1_Load);
((System.ComponentModel.ISupportInitialize)(this.timer1)).EndInit();
this.splitContainer1.Panel1.ResumeLayout(false);
this.splitContainer1.Panel2.ResumeLayout(false);
this.splitContainer1.ResumeLayout(false);
this.dl.ResumeLayout(false);
this.tabPage1.ResumeLayout(false);
this.tabPage1.PerformLayout();
this.tabPage2.ResumeLayout(false);
this.tabPage2.PerformLayout();
# Variables for Clear Image SDK
nsCiFnc.ClearImageFnc Ci;
nsCiFnc.OpStats Op;

# Variables for items to be dumped in the database
private string Name, DOB, SSN, Gender, Height, Weight, Hair, Eye,
Street, City, State, Zip, IDNo, Class, IssueDT, ExpDT, Rest,
Endo, IIN;

# Variables contain the location & name of the image file
[scanned], connection string of the database and path of the
application
private string fileScan, lConnStr, appPath;

private void Form1_Load(object sender, System.EventArgs e)
{
    // Extract the Application's path
    appPath = Directory.GetCurrentDirectory().Replace("\\", "\\\\");

    // Storing the Connection String for the Database
    lConnStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + appPath + "\\\minicheck.mdb;Persist Security Info=True";

    Op = new nsCiFnc.OpStats();
    CmdEnbDis(true);
    // Instantiate ClearImage
    Ci = new nsCiFnc.ClearImageFnc();
    Ci.Op = Op;
}

// Computing Age [returns number of years] using the Date Of
Birth
public int calculateAge(DateTime dob)
{
    int years = DateTime.Now.Year - dob.Year;

    if (DateTime.Now.Month < dob.Month || (DateTime.Now.Month == dob.Month && DateTime.Now.Day < dob.Day))
        years--;

public void dumpInfo()
{
    Name = Name.Trim();
    SSN = SSN.Trim();
    Gender = Gender.Trim();
    Height = Height.Trim();
    Weight = Weight.Trim();
    Hair = Hair.Trim();
    Eye = Eye.Trim();
    Street = Street.Trim();
    City = City.Trim();
    State = State.Trim();
    Zip = Zip.Trim();
    IDNo = IDNo.Trim();
    Class = Class.Trim();
    Rest = Rest.Trim();
    Endo = Endo.Trim();
    IIN = IIN.Trim();

    OleDbConnection conn = new OleDbConnection(lConnStr);

    OleDbDataAdapter cAdapter = new OleDbDataAdapter();

    // Checking for the appropriate query to be executed based on the
    // information available
    if (DOB != null && ExpDT != null && IssueDT != null)
        cAdapter = new OleDbDataAdapter("INSERT INTO
            CUSTOMER_RECORDS (Name1, DOB, SSN, Gender, Height, Weight,
            Hair, Eye, Address1, City, State, Zip, DLNo, Class,
            IssueDT, ExpDT, Rest, Endo, IIN, RecordDate) VALUES ('" +

    if (DOB != null && ExpDT != null && IssueDT == null)
        cAdapter = new OleDbDataAdapter("INSERT INTO
            CUSTOMER_RECORDS (Name1, DOB, SSN, Gender, Height, Weight,
            Hair, Eye, Address1, City, State, Zip, DLNo, Class,
            ExpDT, Rest, Endo, IIN, RecordDate) VALUES ('" + Name + ", " + DOB + ", " + SSN + ", " + Gender + ", " + Height + ", " + Weight + ", " + Hair + ", " + Eye + ", " + Street + ", " + City + ", " + State + ", " + Zip + ", " + IDNo + ", " + Class + ", " + ExpDT + ", " + Rest + ", " + Endo + ", " + IIN + ", " + DateTime.Now + ")", conn);

    if (DOB != null && ExpDT == null && IssueDT != null)
        cAdapter = new OleDbDataAdapter("INSERT INTO
            CUSTOMER_RECORDS (Name1, DOB, SSN, Gender, Height, Weight,
Hair, Eye, Address1, City, State, Zip, DLNo, Class,
IssueDT, Rest, Endo, IIN, RecordDate) VALUES ('" + Name + 
', "+ Date. Time. Parse(DOB) + "+ SSN + "+ Gender + "+ Height + "+ Weight + "+ Hair + "+ Eye + "+ Street + "+ City + "+ State + "+ Zip + "+ IDNo + "+ Class + "+ 
 DateTime. Parse(IssueDT) + "+ Rest + "+ 
 Endo + "+ IIN + "+ DateTime. Now + "+ 
 if (DOB != null && ExpDT != null && IssueDT == null)
cAdapter = new OleDbDataAdapter("INSERT INTO
CUSTOMER_RECORDS (Name1, DOB, SSN, Gender, Height, Weight, 
Hair, Eye, Address1, City, State, Zip, DLNo, Class, Rest, 
Endo, IIN, RecordDate) VALUES ('" + Name + "+ Date. Time. Parse(DOB) + "+ SSN + "+ Gender + "+ Height + "+ Weight + "+ Hair + "+ Eye + "+ Street + "+ City + "+ State + "+ Zip + "+ IDNo + "+ Class + "+ 
 DateTime. Parse(IssueDT) + "+ Rest + "+ 
 Endo + "+ IIN + "+ DateTime. Now + "+ 
 if (DOB == null && ExpDT != null && IssueDT == null)
cAdapter = new OleDbDataAdapter("INSERT INTO
CUSTOMER_RECORDS (Name1, SSN, Gender, Height, Weight, 
Hair, Eye, Address1, City, State, Zip, DLNo, Class, IssueDT, 
ExpDT, Rest, Endo, IIN, RecordDate) VALUES ('" + Name + "+ SSN + "+ Gender + "+ Height + "+ Weight + "+ Hair + "+ Eye + "+ Street + "+ City + "+ State + "+ Zip + "+ IDNo + "+ Class + "+ 
 DateTime. Parse(IssueDT) + "+ ExpDT + "+ Rest + "+ 
 Endo + "+ IIN + "+ DateTime. Now + "+ 
 if (DOB == null && ExpDT == null && IssueDT != null)
cAdapter = new OleDbDataAdapter("INSERT INTO
CUSTOMER_RECORDS (Name1, SSN, Gender, Height, Weight, 
Hair, Eye, Address1, City, State, Zip, DLNo, Class, IssueDT, 
Rest, Endo, IIN, RecordDate) VALUES ('" + Name + "+ SSN + "+ Gender + "+ Height + "+ Weight + "+ Hair + "+ Eye + "+ Street + "+ City + "+ State + "+ Zip + "+ IDNo + "+ Class + "+ 
 DateTime. Parse(IssueDT) + "+ IssueDT + "+ Rest + "+ 
 Endo + "+ IIN + "+ DateTime. Now + "+ 
 if (DOB == null && ExpDT == null && IssueDT == null)
cAdapter = new OleDbDataAdapter("INSERT INTO
CUSTOMER_RECORDS (Name1, SSN, Gender, Height, Weight, 
Hair, Eye, Address1, City, State, Zip, DLNo, Class, IssueDT, 
Rest, Endo, IIN, RecordDate) VALUES ('" + Name + "+ SSN + "+ Gender + "+ Height + "+ Weight + "+ Hair + "+ Eye + "+ Street + "+ State + "+ 
 DateTime. Now + "+ 
if (DOB != null && ExpDT == null && IssueDT != null)
cAdapter = new OleDbDataAdapter("INSERT INTO
CUSTOMER_RECORDS (Name1, DOB, SSN, Gender, Height, Weight, 
Hair, Eye, Address1, City, State, Zip, DLNo, Class, Rest, 
Endo, IIN, RecordDate) VALUES ('" + Name + "+ Date. Time. Parse(DOB) + "+ SSN + "+ Gender + "+ Height + "+ Weight + "+ Hair + "+ Eye + "+ Street + "+ 
 if (DOB == null && ExpDT != null && IssueDT != null)
cAdapter = new OleDbDataAdapter("INSERT INTO
CUSTOMER_RECORDS (Name1, SSN, Gender, Height, Weight, 
Hair, Eye, Address1, City, State, Zip, DLNo, Class, IssueDT, 
Rest, Endo, IIN, RecordDate) VALUES ('" + Name + "+ SSN + "+ Gender + "+ Height + "+ Weight + "+ Hair + "+ Eye + "+ Street + "+ 

if (DOB == null && ExpDT == null && IssueDT == null)
cAdapter = new OleDbDataAdapter("INSERT INTO CUSTOMER_RECORDS (Name1, SSN, Gender, Height, Weight, Hair, Eye, Address1, City, State, Zip, DLNo, Class, Rest, Endo, IIN, RecordDate) VALUES ('" + Name + ", 'SSN + ", 'Gender + ", 'Height + ", 'Weight + ", 'Hair + ", 'Eye + ", 'Street + ", 'City + ", 'State + ", 'Zip + ", 'IDNo + ", 'Class + ", 'Rest + ", 'Endo + ", 'IIN + ", " + DateTime.Now + ", conn);

conn.Open();
DataSet ds = new DataSet();
cAdapter.Fill(ds);
conn.Close();
}

// Magnetic ID coding.
private void bm_Click(object sender, EventArgs e)
{

// Verify if the card is Swiped.
if (tbMagnetic.Text != "")
{
try
{
	bm.Text = "";

Name = "";
DOB = null;
SSN = "";
Gender = "";
Height = "";
Weight = "";
Hair = "";
Eye = "";
Street = "";
City = "";
State = "";
Zip = "";
IDNo = "";
Class = "";
IssueDT = null;
ExpDT = null;
Rest = "";
Endo = "";
IIN = "";

// Split the string based on the specified delimeters
string[] s = tbMagnetic.Text.Split( "?", "^", "%", ";", "=");

tbm.Text = tbm.Text + "State: " + s[1].Substring(0, 2);
State = s[1].Substring(0, 2);
tbm.Text = tbm.Text + "\n\nCity: " + s[1].Substring(2);
City = s[1].Substring(2);
Name = s[2].Replace( "$", " ");
tbm.Text = tbm.Text + "\r\nName: " + Name.Insert(Name.IndexOf(' '), ',');
Name = Name.Insert(Name.IndexOf(' '), ',');
tbm.Text = tbm.Text + "\r\nStreet Address: " + s[3];
Street = s[3];
// Checking if the data contains MagnePrint Signature
if (s[7].Substring(4).ToSequent() == "00000000")
{
    // MagnePrint Enabled Reader
    tbm.Text = tbm.Text + "\r\nExpiration Date: " + s[7].Substring(0, 2) + "/" + s[7].Substring(2, 2);
    tbm.Text = tbm.Text + "\r\nMagnePrint Signature: " + s[8];
}
else
{
    // Regular Reader
    tbm.Text = tbm.Text + "\r\nIssuer Identification Number: " + s[6].Substring(0, 6);
    IIN = s[6].Substring(0, 6);
tbm.Text = tbm.Text + "\r\nID Number: " + s[6].Substring(6);
IDNo = s[6].Substring(6);
tbm.Text = tbm.Text + "\r\nExpiration Date: " + s[7].Substring(0, 2) + "/" + s[7].Substring(2, 2) + "/20" + s[7].Substring(10, 2) + "/20" + s[7].Substring(0, 2);
ExpDT = s[7].Substring(2, 2) + "/" + s[7].Substring(10, 2) + "/20" + s[7].Substring(0, 2);
tbm.Text = tbm.Text + "\r\nDate of Birth: " + s[7].Substring(8, 2) + "/" + s[7].Substring(10, 2) + "/" + s[7].Substring(4, 4);
DOB = s[7].Substring(8, 2) + "/" + s[7].Substring(10, 2) + "/" + s[7].Substring(4, 4);
DateTime dob = DateTime.Parse(s[7].Substring(8, 2) + "/" + s[7].Substring(10, 2) + "/" + s[7].Substring(4, 4));
tbm.Text = tbm.Text + "\r\nAge: " + calculateAge(dob).ToString();
tbm.Text = tbm.Text + "\r\nIssuing Authority: " + s[8].Substring(14, 2);
Class = s[8].Substring(14, 2);
tbm.Text = tbm.Text + "\r\nRestriction: " + s[8].Substring(16, 14);
Rest = s[8].Substring(16, 14);
tbm.Text = tbm.Text + "\r\nGender: " + s[8].Substring(30, 1);
Gender = s[8].Substring(30, 1);
tbm.Text = tbm.Text + "\r\nHeight: " + s[8].Substring(31, 1) + " ft " + s[8].Substring(32, 2) + " inch";
Height = s[8].Substring(31, 1);
tbm.Text = tbm.Text + "\r\nWeight: " + s[8].Substring(34, 3) + " lb";
Weight = s[8].Substring(34, 3);
tbm.Text = tbm.Text + "\nHair Color: " + s[8].Substring(37, 3);
Hair = s[8].Substring(37, 3);
tbm.Text = tbm.Text + "\nEye Color: " + s[8].Substring(40, 3);
Eye = s[8].Substring(40, 3);
}

// Call to dump data in the Database
dumpInfo();

tbMagnetic.Text = "";
bm.Enabled = false;

} catch (Exception E)
{
    // In case of error display on the Interface
    tbMagnetic.Text = "";
tbm.Text = "Error Occurred: " + E.ToString() + "\n\nPlease Swipe the ID again and Parse."
;
}

// Invoking the Scanner Wizard
private void cmdScan_Click(object sender, EventArgs e)
{
    txtRslt.Text = "";
    Process.Start("wiaacmgr.exe");
    bb.Enabled = true;
}

// 2D Barcode Decoding
private void bb_Click(object sender, EventArgs e)
{
    // Fetching the Scanned Image
    string[] openFile = Directory.GetFiles(appPath + "\\", "*.png");
    if (openFile.Length > 0)
        fileScan = openFile[openFile.Length - 1].ToString();

    openFile = Directory.GetFiles(appPath + "\\", "*.tif");
    if (openFile.Length > 0)
        fileScan = openFile[openFile.Length - 1].ToString();

    openFile = Directory.GetFiles(appPath + "\\", "*.bmp");
    if (openFile.Length > 0)
        fileScan = openFile[openFile.Length - 1].ToString();

    openFile = Directory.GetFiles(appPath + "\\", "*.jpg");
    if (openFile.Length > 0)
        fileScan = openFile[openFile.Length - 1].ToString();
// Verify if an Image file was found
if (fileScan != null)
{
    try
    {
        txtRslt.Text = "";

        Name = "";
        DOB = null;
        SSN = "";
        Gender = "";
        Height = "";
        Weight = "";
        Hair = "";
        Eye = "";
        Street = "";
        City = "";
        State = "";
        Zip = "";
        IDNo = "";
        Class = "";
        IssueDT = null;
        ExpDT = null;
        Rest = "";
        Endo = "";
        IIN = "";

        DisplayImage(fileScan);
        if (!OpStart(EOpType.eopPdf417)) return;
        // Do processing
        string s = Ci.ReadPdf417(fileScan);
        DoGC(); // Release ClearImage objects
        // Display results
        txtRslt.Text = "The RAW data:\n\n" + s;

        // Split the string based on the line feed and carriage return characters
        return characters
        string[] n = s.Split(\r\n', 'n');

        txtRslt.Text = txtRslt.Text + \r\nThe Parsed Result:\n\n" + s;

        // Loop through the substrings and check for starting Identifiers [first 3 characters]
        foreach (string t in n)
        {
            if (t.StartsWith("DAQ"))
            {
                txtRslt.Text = txtRslt.Text + \r\nID Number: " + t.Substring(3);
                IDNo = t.Substring(3);
            }
            if (t.StartsWith("DAA") || t.StartsWith("DBN"))
            {
                txtRslt.Text = txtRslt.Text + \r\nFull Name: " + t.Substring(3);
                Name = t.Substring(3);
            }
        }
    }
}
if (t.StartsWith("DAB") || t.StartsWith("DBO"))
{
    txtRslt.Text = txtRslt.Text + "\nLast Name: " +
    t.Substring(3);
    Name = t.Substring(3) + " " + Name;
}
if (t.StartsWith("DAC") || t.StartsWith("DBP"))
{
    txtRslt.Text = txtRslt.Text + "\nFirst Name: " +
    t.Substring(3);
    Name = Name + t.Substring(3);
}
if (t.StartsWith("DAD") || t.StartsWith("DBQ"))
{
    txtRslt.Text = txtRslt.Text + "\nMiddle Name: "+
    t.Substring(3);
    Name = Name + t.Substring(3);
}
if (t.StartsWith("DAG") || t.StartsWith("DAH") ||
    t.StartsWith("DAL") || t.StartsWith("DAM"))
{
    txtRslt.Text = txtRslt.Text + "\nStreet Address:
    " + t.Substring(3);
    Street = t.Substring(3);
}
if (t.StartsWith("DAI") || t.StartsWith("DAN"))
{
    txtRslt.Text = txtRslt.Text + "\nCity: " +
    t.Substring(3);
    City = t.Substring(3);
}
if (t.StartsWith("DAJ") || t.StartsWith("DAO"))
{
    txtRslt.Text = txtRslt.Text + "\nState: " +
    t.Substring(3);
    State = t.Substring(3);
}
if (t.StartsWith("DAK") || t.StartsWith("DAP"))
{
    txtRslt.Text = txtRslt.Text + "\nZip Code: " +
    t.Substring(3);
    Zip = t.Substring(3);
}
if (t.StartsWith("DAR") || t.StartsWith("PAA"))
{
    txtRslt.Text = txtRslt.Text + "\nLicense Class:
    " + t.Substring(3);
    Class = t.Substring(3);
}
if (t.StartsWith("DAU") || t.StartsWith("DAV"))
{
    txtRslt.Text = txtRslt.Text + "\nHeight: " +
    t.Substring(3, 1) + "ft " + t.Substring(4) +
    "inch";
    Height = t.Substring(3);
}
if (t.StartsWith("DAW") || t.StartsWith("DAX"))
{
    txtRslt.Text = txtRslt.Text + "\nWeight: " +
    t.Substring(3);
    Weight = t.Substring(3);
}
if (t.StartsWith("DAY"))
{
    txtRslt.Text = txtRslt.Text + "\nEye Color: " +
    t.Substring(3);
    Eye = t.Substring(3);
}
if (t.StartsWith("DAZ"))
{
    txtRslt.Text = txtRslt.Text + "\nHair Color: " +
    t.Substring(3);
    Hair = t.Substring(3);
}
if (t.StartsWith("DBA") || t.StartsWith("PAB"))
{
    txtRslt.Text = txtRslt.Text + "\nExpiration 
Date: " + t.Substring(7, 2) + "/" +
    t.Substring(9, 2) + "/" + t.Substring(3, 4);
    ExpDT = t.Substring(7, 2) + "/" + t.Substring(9,
    2) + "/" + t.Substring(3, 4);
}
if (t.StartsWith("DBB"))
{
    txtRslt.Text = txtRslt.Text + "\nDate of Birth: 
" + t.Substring(7, 2) + "/" + t.Substring(9, 2) +
    "/" + t.Substring(3, 4);
    DOB = t.Substring(7, 2) + "/" + t.Substring(9, 2) +
    "/" + t.Substring(3, 4);
    DateTime dob = DateTime.Parse(t.Substring(7, 2) +
    "/" + t.Substring(9, 2) + "/" + t.Substring(3,
    4));
    txtRslt.Text = txtRslt.Text + "\nAge: " +
calculateAge(dob).ToString();
}
if (t.StartsWith("DBC"))
{
    txtRslt.Text = txtRslt.Text + "\nGender: " +
t.Substring(3);
    Gender = t.Substring(3);
}
if (t.StartsWith("DBD") || t.StartsWith("PAD"))
{
    txtRslt.Text = txtRslt.Text + "\nDate of Issue:
" + t.Substring(7, 2) + "/" + t.Substring(9, 2) +
    "/" + t.Substring(3, 4);
    IssueDT = t.Substring(7, 2) + "/" +
    t.Substring(9, 2) + "/" + t.Substring(3, 4);
if (t.StartsWith("DAS") || t.StartsWith("PAE"))
{
    txtRslt.Text = txtRslt.Text + "\n\nRestriction: " + t.Substring(3);
    Rest = t.Substring(3);
}
if (t.StartsWith("DAT") || t.StartsWith("PAF"))
{
    txtRslt.Text = txtRslt.Text + "\n\nEndorsement: " + t.Substring(3);
    Endo = t.Substring(3);
}
if (t.StartsWith("DBK") || t.StartsWith("DBM"))
{
    txtRslt.Text = txtRslt.Text + "\n\nSSN: " + t.Substring(3);
    SSN = t.Substring(3);
}
//txtRslt.Text = txtRslt.Text + "\n" + t;
}

// Deleting the saved Image(s)
string[] delFile = Directory.GetFiles(appPath + "\\", "*.jpg");
foreach (string file in delFile)
{
    File.Delete(file);
}
delFile = Directory.GetFiles(appPath + "\\", "*.bmp");
foreach (string file in delFile)
{
    File.Delete(file);
}
delFile = Directory.GetFiles(appPath + "\\", "*.tif");
foreach (string file in delFile)
{
    File.Delete(file);
}
delFile = Directory.GetFiles(appPath + "\\", "*.png");
foreach (string file in delFile)
{
    File.Delete(file);
}

fileScan = null;
bb.Enabled = false;

// Dump information in the Database
dumpInfo();

} catch (Exception E)
{
    // Display errors in the interface
txtRslt.Text = "Error Occured: " + E.ToString() + "\n\nPlease Scan the ID again and Parse.\n";
// Buttons to Show and Hide tab panels
private void bMagnetic_Click(object sender, EventArgs e)
{
    tabPage1.Show();
    tabPage2.Hide();
    tabPage3.Hide();
    tabPage4.Hide();

    tbMagnetic.Focus();
}

private void bBarcode_Click(object sender, EventArgs e)
{
    tabPage2.Show();
    tabPage1.Hide();
    tabPage3.Hide();
    tabPage4.Hide();
}

private void bMagnePrint_Click(object sender, EventArgs e)
{
    tabPage3.Show();
    tabPage4.Hide();
    tabPage2.Hide();
    tabPage1.Hide();
}

private void bReport_Click(object sender, EventArgs e)
{
    tabPage4.Show();
    tabPage3.Hide();
    tabPage2.Hide();
    tabPage1.Hide();
}

private void tbMagnetic_Enter(object sender, EventArgs e)
{
    tbm.Text = "";
    bm.Enabled = true;
}

// The code starting from this point was part of the Demo Application that came with Clear Image SDK.
protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose(disposing);
}
private void DoGC()
{
    System.GC.Collect();
    System.GC.WaitForPendingFinalizers();
}

private bool OpStart(EOpType nType)
{
    txtRslt.Text = "";
    if ((fileScan == "")) { MessageBox.Show("No File specified");
    return false; }
    DisplayImage(fileScan);
    Op.Start(nType);
    return true;
}

//###########################################################################
#
//  Image file browsing and display
#
//###########################################################################
#
private bool GetThumbnailImageAbort()
{
    return false;
}

private void DisplayImage(string sFile)
{
    try
    {
        PictureBox1.Image = null;
        if (!System.IO.File.Exists(sFile)) return;
        if (System.IO.Path.GetExtension(sFile).ToLower() == @".pdf") return;
        Image newImage = Image.FromFile(sFile);
        double scaleX = (double)PictureBox1.Width / (double)newImage.Width;
        double scaleY = (double)PictureBox1.Height / (double)newImage.Height;
        double Scale = scaleX;
        if ((Scale > scaleY))
        {
            Scale = scaleY;
        }
        int w;
        int h;
        w = (int)(newImage.Width * Scale);
        h = (int)(newImage.Height * Scale);
        PictureBox1.Image = newImage.GetThumbnailImage(w, h, new
System.Drawing.Image.GetThumbnailImageAbort(GetThumbnailImageAbort), IntPtr.Zero);
    }
}
catch (Exception ex)
{
    ShowError(ex);
}

private void ShowError(Exception ex)
{
    MessageBox.Show("ERROR: " + ex.ToString());
}

private void CmdEnbDis(bool bEnb)
{
    cmdScan.Enabled = bEnb;
    timer1.Enabled = false;
}

private bool OpTest(long nReportStep)
{
    string s = "";
    s = Ci.ReadPdf417(fileScan);

    // Release ClearImage objects
    DoGC();
    // Check for error
    if (s.StartsWith("ERROR"))
    {
        txtRslt.Text = s;
        return false;
    }
    // Update display every 10 runs
    if ((Op.cnt == 1) || ((Op.cnt % nReportStep) == 0))
    {
        txtRslt.Text = s;
    }
    return true;
}

private void timer1_Elapsed(object sender, System.Timers.ElapsedEventArgs e)
{
    timer1.Enabled = false; // Prevent reentrancy
    if (!OpTest(1))
    {
        CmdEnbDis(true); // On error
    }
    else
    {
        timer1.Enabled = true;
    }
}