Form : CaptureTest

```
using System;
using System.Diagnostics;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using DirectX.Capture;
using Microsoft.VisualBasic;
using AviFile;
using AXmsCtrl;
using ASmsCtrl;

namespace CaptureTest1
{
      public class CaptureTest1 : System.Windows.Forms.Form
      {
        public int video_index = 1;
        public int audio_index = 4;
        public string messageSender = "";

        private Capture capture = null;
            private Filters filters = new Filters();


            private System.Windows.Forms.TextBox txtFilename;
        private System.Windows.Forms.Label label1;
            private System.Windows.Forms.MenuItem menuItem1;
            private System.Windows.Forms.MenuItem menuItem7;
            private System.Windows.Forms.MainMenu mainMenu;
            private System.Windows.Forms.MenuItem mnuExit;
            private System.Windows.Forms.MenuItem mnuDevices;
            private System.Windows.Forms.MenuItem
mnuVideoDevices;
        private System.Windows.Forms.MenuItem mnuAudioDevices;
        private System.Windows.Forms.Panel panelVideo;
        private System.Windows.Forms.Button btnCue;
            private System.Windows.Forms.MenuItem mnuPreview;
            private System.Windows.Forms.MenuItem menuItem8;
        private System.Windows.Forms.MenuItem mnuPropertyPages;
        private Timer timer1;
        private Label labelTimer;
        private Label label2;
        private TextBox textBoxTimer;
        private Label label3;
        private Label label4;
        private Label label5;
        private Button buttonCapture;
        private IContainer components;
        private MmsProtocolMm1 objMm1Protocol;
        private MmsConstants objMmsConstants;
        private MenuItem menuItem2;
        private MenuItem menuItem3;
        private Timer timer2;
        private ComboBox comboBoxHPDev;
        private TextBox textBoxLogfile;
        private TextBox textBoxResult;
```

```
        private Label label6;
        private Label label7;
        private Label label8;
        internal TextBox ctlServerAddress;
        internal TextBox ctlServerPassword;
        internal TextBox ctlServerLogin;
        internal TextBox ctlServerGateway;
        internal TextBox ctlServerAPN;
        internal Label label9;
        internal Label label10;
        internal Label label11;
        internal Label label12;
        internal Label label13;
        private CaptureTest.DS_Webcam dS_Webcam;
        private BindingSource tableNoHPBindingSource;
        private
global::CaptureTest.DS_WebcamTableAdapters.Table_NoHPTableAdapter
table_NoHPTableAdapter;
        private Button button1;
        private Label btnExit;
        private Label btnStop;
        private Label btnStart;
        private Label button2;
        private Label label14;
        private Label label15;
        private Label label16;
        private Label label17;
        private Label label18;
        private Label label19;
        private Label label20;
        private Label label21;
        private Label label22;
        private Label label23;
        private Label label24;
        private Timer timer3;
        private GsmIn objGsmIn = new GsmIn();


            public CaptureTest1()
            {
                  InitializeComponent();

                  #if DEBUG
                  capture = new Capture(
filters.VideoInputDevices[1], filters.AudioInputDevices[4] );
                  capture.CaptureComplete += new EventHandler(
OnCaptureComplete );
                  #endif

            objMm1Protocol = new MmsProtocolMm1();
            objMmsConstants = new MmsConstants();

                  try { updateMenu(); } catch {}
            }

            /// <summary>
            /// Clean up any resources being used.
```

```
            /// </summary>
            protected override void Dispose( bool disposing )
            {
                  if( disposing )
                  {
                        if(components != null)
                        {
                              components.Dispose();
                        }
                  }
                  base.Dispose( disposing );
            }

            #region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not
modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
            this.components = new
System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager
resources = new
System.ComponentModel.ComponentResourceManager(typeof(CaptureTest
1));
            this.txtFilename = new
System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.mainMenu = new
System.Windows.Forms.MainMenu(this.components);
            this.menuItem1 = new System.Windows.Forms.MenuItem();
            this.menuItem2 = new System.Windows.Forms.MenuItem();
            this.menuItem3 = new System.Windows.Forms.MenuItem();
            this.mnuExit = new System.Windows.Forms.MenuItem();
            this.mnuDevices = new
System.Windows.Forms.MenuItem();
            this.mnuVideoDevices = new
System.Windows.Forms.MenuItem();
            this.mnuAudioDevices = new
System.Windows.Forms.MenuItem();
            this.menuItem7 = new System.Windows.Forms.MenuItem();
            this.mnuPropertyPages = new
System.Windows.Forms.MenuItem();
            this.menuItem8 = new System.Windows.Forms.MenuItem();
            this.mnuPreview = new
System.Windows.Forms.MenuItem();
            this.panelVideo = new System.Windows.Forms.Panel();
            this.button1 = new System.Windows.Forms.Button();
            this.btnCue = new System.Windows.Forms.Button();
            this.timer1 = new
System.Windows.Forms.Timer(this.components);
            this.labelTimer = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.textBoxTimer = new
System.Windows.Forms.TextBox();
```

```
            this.label3 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.buttonCapture = new
System.Windows.Forms.Button();
            this.timer2 = new
System.Windows.Forms.Timer(this.components);
            this.comboBoxHPDev = new
System.Windows.Forms.ComboBox();
            this.textBoxLogfile = new
System.Windows.Forms.TextBox();
            this.textBoxResult = new
System.Windows.Forms.TextBox();
            this.label6 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.ctlServerAddress = new
System.Windows.Forms.TextBox();
            this.ctlServerPassword = new
System.Windows.Forms.TextBox();
            this.ctlServerLogin = new
System.Windows.Forms.TextBox();
            this.ctlServerGateway = new
System.Windows.Forms.TextBox();
            this.ctlServerAPN = new
System.Windows.Forms.TextBox();
            this.label9 = new System.Windows.Forms.Label();
            this.label10 = new System.Windows.Forms.Label();
            this.label11 = new System.Windows.Forms.Label();
            this.label12 = new System.Windows.Forms.Label();
            this.label13 = new System.Windows.Forms.Label();
            this.dS_Webcam = new CaptureTest.DS_Webcam();
            this.tableNoHPBindingSource = new
System.Windows.Forms.BindingSource(this.components);
            this.table_NoHPTableAdapter = new
CaptureTest.DS_WebcamTableAdapters.Table_NoHPTableAdapter();
            this.btnExit = new System.Windows.Forms.Label();
            this.btnStop = new System.Windows.Forms.Label();
            this.btnStart = new System.Windows.Forms.Label();
            this.button2 = new System.Windows.Forms.Label();
            this.label14 = new System.Windows.Forms.Label();
            this.label15 = new System.Windows.Forms.Label();
            this.label16 = new System.Windows.Forms.Label();
            this.label17 = new System.Windows.Forms.Label();
            this.label18 = new System.Windows.Forms.Label();
            this.label19 = new System.Windows.Forms.Label();
            this.label20 = new System.Windows.Forms.Label();
            this.label21 = new System.Windows.Forms.Label();
            this.label22 = new System.Windows.Forms.Label();
            this.label23 = new System.Windows.Forms.Label();
            this.label24 = new System.Windows.Forms.Label();
            this.timer3 = new
System.Windows.Forms.Timer(this.components);

((System.ComponentModel.ISupportInitialize)(this.dS_Webcam)).Begi
nInit();
```

```
((System.ComponentModel.ISupportInitialize)(this.tableNoHPBinding
Source)).BeginInit();
            this.SuspendLayout();
            //
            // txtFilename
            //
            this.txtFilename.Location = new
System.Drawing.Point(437, 138);
            this.txtFilename.Name = "txtFilename";
            this.txtFilename.Size = new System.Drawing.Size(181,
20);
            this.txtFilename.TabIndex = 0;
            this.txtFilename.Text = "c:\\test.avi";
            //
            // label1
            //
            this.label1.BackColor =
System.Drawing.Color.Transparent;
            this.label1.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label1.Location = new System.Drawing.Point(366,
141);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(64, 16);
            this.label1.TabIndex = 1;
            this.label1.Text = "Filename";
            //
            // mainMenu
            //
            this.mainMenu.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.menuItem1,
            this.mnuDevices,
            this.menuItem7});
            //
            // menuItem1
            //
            this.menuItem1.Index = 0;
            this.menuItem1.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.menuItem2,
            this.menuItem3,
            this.mnuExit});
            this.menuItem1.Text = "File";
            //
            // menuItem2
            //
            this.menuItem2.Index = 0;
            this.menuItem2.Text = "Handphone Number";
            this.menuItem2.Click += new
System.EventHandler(this.menuItem2_Click);
            //
            // menuItem3
            //
            this.menuItem3.Index = 1;
            this.menuItem3.Text = "-";
```

```
            //
            // mnuExit
            //
            this.mnuExit.Index = 2;
            this.mnuExit.Text = "E&xit";
            this.mnuExit.Click += new
System.EventHandler(this.mnuExit_Click);
            //
            // mnuDevices
            //
            this.mnuDevices.Index = 1;
            this.mnuDevices.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.mnuVideoDevices,
            this.mnuAudioDevices});
            this.mnuDevices.Text = "Devices";
            //
            // mnuVideoDevices
            //
            this.mnuVideoDevices.Index = 0;
            this.mnuVideoDevices.Text = "Video Devices";
            //
            // mnuAudioDevices
            //
            this.mnuAudioDevices.Index = 1;
            this.mnuAudioDevices.Text = "Audio Devices";
            //
            // menuItem7
            //
            this.menuItem7.Index = 2;
            this.menuItem7.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.mnuPropertyPages,
            this.menuItem8,
            this.mnuPreview});
            this.menuItem7.Text = "Options";
            //
            // mnuPropertyPages
            //
            this.mnuPropertyPages.Index = 0;
            this.mnuPropertyPages.Text = "PropertyPages";
            this.mnuPropertyPages.Click += new
System.EventHandler(this.mnuPropertyPages_Click);
            //
            // menuItem8
            //
            this.menuItem8.Index = 1;
            this.menuItem8.Text = "-";
            this.menuItem8.Visible = false;
            //
            // mnuPreview
            //
            this.mnuPreview.Index = 2;
            this.mnuPreview.Text = "Preview";
            this.mnuPreview.Click += new
System.EventHandler(this.mnuPreview_Click);
            //
```

```
            // panelVideo
            //
            this.panelVideo.BackColor =
System.Drawing.Color.Transparent;
            this.panelVideo.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.panelVideo.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.panelVideo.Location = new
System.Drawing.Point(12, 58);
            this.panelVideo.Name = "panelVideo";
            this.panelVideo.Size = new System.Drawing.Size(343,
257);
            this.panelVideo.TabIndex = 6;
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(109,
168);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(94, 52);
            this.button1.TabIndex = 32;
            this.button1.Text = "convert1";
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // btnCue
            //
            this.btnCue.Location = new System.Drawing.Point(244,
125);
            this.btnCue.Name = "btnCue";
            this.btnCue.Size = new System.Drawing.Size(80, 24);
            this.btnCue.TabIndex = 8;
            this.btnCue.Text = "Cue";
            this.btnCue.Visible = false;
            this.btnCue.Click += new
System.EventHandler(this.btnCue_Click);
            //
            // timer1
            //
            this.timer1.Interval = 1000;
            this.timer1.Tick += new
System.EventHandler(this.timer1_Tick);
            //
            // labelTimer
            //
            this.labelTimer.BackColor =
System.Drawing.Color.Transparent;
            this.labelTimer.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.labelTimer.Location = new
System.Drawing.Point(437, 188);
            this.labelTimer.Name = "labelTimer";
            this.labelTimer.Size = new System.Drawing.Size(35,
16);
            this.labelTimer.TabIndex = 9;
            this.labelTimer.Text = "0";
```

```
            //
            // label2
            //
            this.label2.BackColor =
System.Drawing.Color.Transparent;
            this.label2.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label2.Location = new System.Drawing.Point(366,
167);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(64, 16);
            this.label2.TabIndex = 11;
            this.label2.Text = "Timer";
            //
            // textBoxTimer
            //
            this.textBoxTimer.Location = new
System.Drawing.Point(437, 164);
            this.textBoxTimer.Name = "textBoxTimer";
            this.textBoxTimer.Size = new System.Drawing.Size(29,
20);
            this.textBoxTimer.TabIndex = 10;
            this.textBoxTimer.Text = "5";
            //
            // label3
            //
            this.label3.BackColor =
System.Drawing.Color.Transparent;
            this.label3.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label3.Location = new System.Drawing.Point(470,
166);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(50, 16);
            this.label3.TabIndex = 12;
            this.label3.Text = "second";
            //
            // label4
            //
            this.label4.BackColor =
System.Drawing.Color.Transparent;
            this.label4.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label4.Location = new System.Drawing.Point(366,
188);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(64, 16);
            this.label4.TabIndex = 13;
            this.label4.Text = "Running ";
            //
            // label5
            //
            this.label5.BackColor =
System.Drawing.Color.Transparent;
            this.label5.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
```

```
            this.label5.Location = new System.Drawing.Point(470,
187);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(50, 16);
            this.label5.TabIndex = 14;
            this.label5.Text = "second";
            //
            // buttonCapture
            //
            this.buttonCapture.Location = new
System.Drawing.Point(217, 121);
            this.buttonCapture.Name = "buttonCapture";
            this.buttonCapture.Size = new System.Drawing.Size(80,
24);
            this.buttonCapture.TabIndex = 15;
            this.buttonCapture.Text = "Capture";
            this.buttonCapture.Visible = false;
            this.buttonCapture.Click += new
System.EventHandler(this.buttonCapture_Click);
            //
            // timer2
            //
            this.timer2.Interval = 10000;
            this.timer2.Tick += new
System.EventHandler(this.timer2_Tick);
            //
            // comboBoxHPDev
            //
            this.comboBoxHPDev.FormattingEnabled = true;
            this.comboBoxHPDev.Location = new
System.Drawing.Point(437, 59);
            this.comboBoxHPDev.Name = "comboBoxHPDev";
            this.comboBoxHPDev.Size = new
System.Drawing.Size(181, 21);
            this.comboBoxHPDev.TabIndex = 16;
            //
            // textBoxLogfile
            //
            this.textBoxLogfile.Location = new
System.Drawing.Point(437, 86);
            this.textBoxLogfile.Name = "textBoxLogfile";
            this.textBoxLogfile.Size = new
System.Drawing.Size(181, 20);
            this.textBoxLogfile.TabIndex = 17;
            this.textBoxLogfile.Text = "c:\\logfile.txt";
            //
            // textBoxResult
            //
            this.textBoxResult.Location = new
System.Drawing.Point(437, 112);
            this.textBoxResult.Name = "textBoxResult";
            this.textBoxResult.Size = new
System.Drawing.Size(181, 20);
            this.textBoxResult.TabIndex = 18;
            //
            // label6
            //
```

```
            this.label6.BackColor =
System.Drawing.Color.Transparent;
            this.label6.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label6.Location = new System.Drawing.Point(366,
115);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(64, 16);
            this.label6.TabIndex = 21;
            this.label6.Text = "Result ";
            //
            // label7
            //
            this.label7.BackColor =
System.Drawing.Color.Transparent;
            this.label7.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label7.Location = new System.Drawing.Point(366,
89);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(64, 16);
            this.label7.TabIndex = 20;
            this.label7.Text = "LogFile ";
            //
            // label8
            //
            this.label8.BackColor =
System.Drawing.Color.Transparent;
            this.label8.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label8.Location = new System.Drawing.Point(366,
56);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(64, 31);
            this.label8.TabIndex = 19;
            this.label8.Text = "Receiver Devices";
            //
            // ctlServerAddress
            //
            this.ctlServerAddress.Location = new
System.Drawing.Point(369, 29);
            this.ctlServerAddress.Name = "ctlServerAddress";
            this.ctlServerAddress.Size = new
System.Drawing.Size(249, 20);
            this.ctlServerAddress.TabIndex = 31;
            this.ctlServerAddress.Text =
"http://mmc.xl.net.id/servlets/mms";
            //
            // ctlServerPassword
            //
            this.ctlServerPassword.Location = new
System.Drawing.Point(534, 5);
            this.ctlServerPassword.Name = "ctlServerPassword";
            this.ctlServerPassword.PasswordChar = '*';
            this.ctlServerPassword.Size = new
System.Drawing.Size(84, 20);
            this.ctlServerPassword.TabIndex = 27;
```

```
            this.ctlServerPassword.Text = "proxl";
            //
            // ctlServerLogin
            //
            this.ctlServerLogin.Location = new
System.Drawing.Point(369, 5);
            this.ctlServerLogin.Name = "ctlServerLogin";
            this.ctlServerLogin.Size = new
System.Drawing.Size(97, 20);
            this.ctlServerLogin.TabIndex = 25;
            this.ctlServerLogin.Text = "xlgprs";
            //
            // ctlServerGateway
            //
            this.ctlServerGateway.Location = new
System.Drawing.Point(76, 29);
            this.ctlServerGateway.Name = "ctlServerGateway";
            this.ctlServerGateway.Size = new
System.Drawing.Size(235, 20);
            this.ctlServerGateway.TabIndex = 29;
            this.ctlServerGateway.Text = "202.152.240.50";
            //
            // ctlServerAPN
            //
            this.ctlServerAPN.Location = new
System.Drawing.Point(76, 5);
            this.ctlServerAPN.Name = "ctlServerAPN";
            this.ctlServerAPN.Size = new System.Drawing.Size(235,
20);
            this.ctlServerAPN.TabIndex = 23;
            this.ctlServerAPN.Text = "www.xlmms.net";
            //
            // label9
            //
            this.label9.BackColor =
System.Drawing.Color.Transparent;
            this.label9.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label9.Location = new System.Drawing.Point(320,
33);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(100, 23);
            this.label9.TabIndex = 30;
            this.label9.Text = "S&erver";
            //
            // label10
            //
            this.label10.BackColor =
System.Drawing.Color.Transparent;
            this.label10.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label10.Location = new System.Drawing.Point(12,
33);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(56, 16);
            this.label10.TabIndex = 28;
            this.label10.Text = "&Gateway";
```

```
            //
            // label11
            //
            this.label11.BackColor =
System.Drawing.Color.Transparent;
            this.label11.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label11.Location = new System.Drawing.Point(470,
9);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(100, 23);
            this.label11.TabIndex = 26;
            this.label11.Text = "&Password";
            //
            // label12
            //
            this.label12.BackColor =
System.Drawing.Color.Transparent;
            this.label12.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label12.Location = new System.Drawing.Point(320,
9);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(56, 16);
            this.label12.TabIndex = 24;
            this.label12.Text = "L&ogin";
            //
            // label13
            //
            this.label13.BackColor =
System.Drawing.Color.Transparent;
            this.label13.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label13.Location = new System.Drawing.Point(12,
9);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(40, 16);
            this.label13.TabIndex = 22;
            this.label13.Text = "&APN";
            //
            // dS_Webcam
            //
            this.dS_Webcam.DataSetName = "DS_Webcam";
            this.dS_Webcam.SchemaSerializationMode =
System.Data.SchemaSerializationMode.IncludeSchema;
            //
            // tableNoHPBindingSource
            //
            this.tableNoHPBindingSource.DataMember =
"Table_NoHP";
            this.tableNoHPBindingSource.DataSource =
this.dS_Webcam;
            //
            // table_NoHPTableAdapter
            //
            this.table_NoHPTableAdapter.ClearBeforeFill = true;
            //
```

```
            // btnExit
            //
            this.btnExit.BackColor =
System.Drawing.Color.Transparent;
            this.btnExit.Cursor =
System.Windows.Forms.Cursors.Hand;
            this.btnExit.Font = new System.Drawing.Font("Tahoma",
9.75F, System.Drawing.FontStyle.Bold);
            this.btnExit.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.btnExit.Image =
((System.Drawing.Image)(resources.GetObject("btnExit.Image")));
            this.btnExit.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.btnExit.Location = new System.Drawing.Point(563,
246);
            this.btnExit.Name = "btnExit";
            this.btnExit.Size = new System.Drawing.Size(55, 70);
            this.btnExit.TabIndex = 36;
            this.btnExit.Text = " ";
            this.btnExit.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.btnExit.Click += new
System.EventHandler(this.btnExit_Click);
            //
            // btnStop
            //
            this.btnStop.BackColor =
System.Drawing.Color.Transparent;
            this.btnStop.Cursor =
System.Windows.Forms.Cursors.Hand;
            this.btnStop.Font = new System.Drawing.Font("Tahoma",
9.75F, System.Drawing.FontStyle.Bold);
            this.btnStop.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.btnStop.Image =
((System.Drawing.Image)(resources.GetObject("btnStop.Image")));
            this.btnStop.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.btnStop.Location = new System.Drawing.Point(473,
246);
            this.btnStop.Name = "btnStop";
            this.btnStop.Size = new System.Drawing.Size(55, 70);
            this.btnStop.TabIndex = 35;
            this.btnStop.Text = " ";
            this.btnStop.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.btnStop.Click += new
System.EventHandler(this.btnStop_Click);
            //
            // btnStart
            //
            this.btnStart.BackColor =
System.Drawing.Color.Transparent;
            this.btnStart.Cursor =
System.Windows.Forms.Cursors.Hand;
```

```
            this.btnStart.Font = new
System.Drawing.Font("Tahoma", 9.75F,
System.Drawing.FontStyle.Bold);
            this.btnStart.ForeColor = System.Drawing.Color.Black;
            this.btnStart.Image =
((System.Drawing.Image)(resources.GetObject("btnStart.Image")));
            this.btnStart.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.btnStart.Location = new
System.Drawing.Point(376, 246);
            this.btnStart.Name = "btnStart";
            this.btnStart.Size = new System.Drawing.Size(55, 70);
            this.btnStart.TabIndex = 34;
            this.btnStart.Text = " ";
            this.btnStart.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.btnStart.Click += new
System.EventHandler(this.btnStart_Click);
            //
            // button2
            //
            this.button2.BackColor =
System.Drawing.Color.Transparent;
            this.button2.Cursor =
System.Windows.Forms.Cursors.Hand;
            this.button2.Image =
((System.Drawing.Image)(resources.GetObject("button2.Image")));
            this.button2.Location = new System.Drawing.Point(497,
211);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(121, 28);
            this.button2.TabIndex = 0;
            this.button2.Click += new
System.EventHandler(this.button2_Click);
            //
            // label14
            //
            this.label14.AutoSize = true;
            this.label14.BackColor =
System.Drawing.Color.Transparent;
            this.label14.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label14.Location = new System.Drawing.Point(62,
8);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(10, 13);
            this.label14.TabIndex = 0;
            this.label14.Text = ":";
            //
            // label15
            //
            this.label15.AutoSize = true;
            this.label15.BackColor =
System.Drawing.Color.Transparent;
            this.label15.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
```

```
            this.label15.Location = new System.Drawing.Point(62,
32);
            this.label15.Name = "label15";
            this.label15.Size = new System.Drawing.Size(10, 13);
            this.label15.TabIndex = 0;
            this.label15.Text = ":";
            //
            // label16
            //
            this.label16.AutoSize = true;
            this.label16.BackColor =
System.Drawing.Color.Transparent;
            this.label16.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label16.Location = new System.Drawing.Point(356,
8);
            this.label16.Name = "label16";
            this.label16.Size = new System.Drawing.Size(10, 13);
            this.label16.TabIndex = 37;
            this.label16.Text = ":";
            //
            // label17
            //
            this.label17.AutoSize = true;
            this.label17.BackColor =
System.Drawing.Color.Transparent;
            this.label17.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label17.Location = new System.Drawing.Point(356,
32);
            this.label17.Name = "label17";
            this.label17.Size = new System.Drawing.Size(10, 13);
            this.label17.TabIndex = 37;
            this.label17.Text = ":";
            //
            // label18
            //
            this.label18.AutoSize = true;
            this.label18.BackColor =
System.Drawing.Color.Transparent;
            this.label18.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label18.Location = new System.Drawing.Point(523,
9);
            this.label18.Name = "label18";
            this.label18.Size = new System.Drawing.Size(10, 13);
            this.label18.TabIndex = 38;
            this.label18.Text = ":";
            //
            // label19
            //
            this.label19.AutoSize = true;
            this.label19.BackColor =
System.Drawing.Color.Transparent;
            this.label19.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
```

```
            this.label19.Location = new System.Drawing.Point(423,
62);
            this.label19.Name = "label19";
            this.label19.Size = new System.Drawing.Size(10, 13);
            this.label19.TabIndex = 39;
            this.label19.Text = ":";
            //
            // label20
            //
            this.label20.AutoSize = true;
            this.label20.BackColor =
System.Drawing.Color.Transparent;
            this.label20.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label20.Location = new System.Drawing.Point(423,
89);
            this.label20.Name = "label20";
            this.label20.Size = new System.Drawing.Size(10, 13);
            this.label20.TabIndex = 39;
            this.label20.Text = ":";
            //
            // label21
            //
            this.label21.AutoSize = true;
            this.label21.BackColor =
System.Drawing.Color.Transparent;
            this.label21.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label21.Location = new System.Drawing.Point(423,
115);
            this.label21.Name = "label21";
            this.label21.Size = new System.Drawing.Size(10, 13);
            this.label21.TabIndex = 39;
            this.label21.Text = ":";
            //
            // label22
            //
            this.label22.AutoSize = true;
            this.label22.BackColor =
System.Drawing.Color.Transparent;
            this.label22.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label22.Location = new System.Drawing.Point(423,
141);
            this.label22.Name = "label22";
            this.label22.Size = new System.Drawing.Size(10, 13);
            this.label22.TabIndex = 39;
            this.label22.Text = ":";
            //
            // label23
            //
            this.label23.AutoSize = true;
            this.label23.BackColor =
System.Drawing.Color.Transparent;
            this.label23.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
```

```
            this.label23.Location = new System.Drawing.Point(423,
167);
            this.label23.Name = "label23";
            this.label23.Size = new System.Drawing.Size(10, 13);
            this.label23.TabIndex = 39;
            this.label23.Text = ":";
            //
            // label24
            //
            this.label24.AutoSize = true;
            this.label24.BackColor =
System.Drawing.Color.Transparent;
            this.label24.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label24.Location = new System.Drawing.Point(423,
187);
            this.label24.Name = "label24";
            this.label24.Size = new System.Drawing.Size(10, 13);
            this.label24.TabIndex = 39;
            this.label24.Text = ":";
            //
            // timer3
            //
            this.timer3.Enabled = true;
            this.timer3.Interval = 1000;
            this.timer3.Tick += new
System.EventHandler(this.timer3_Tick);
            //
            // CaptureTest1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5,
13);
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.BackgroundImag
e")));
            this.ClientSize = new System.Drawing.Size(630, 324);
            this.Controls.Add(this.label24);
            this.Controls.Add(this.label23);
            this.Controls.Add(this.label22);
            this.Controls.Add(this.label21);
            this.Controls.Add(this.label20);
            this.Controls.Add(this.label19);
            this.Controls.Add(this.label17);
            this.Controls.Add(this.label16);
            this.Controls.Add(this.label15);
            this.Controls.Add(this.label14);
            this.Controls.Add(this.labelTimer);
            this.Controls.Add(this.textBoxResult);
            this.Controls.Add(this.textBoxLogfile);
            this.Controls.Add(this.comboBoxHPDev);
            this.Controls.Add(this.textBoxTimer);
            this.Controls.Add(this.txtFilename);
            this.Controls.Add(this.panelVideo);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.btnExit);
            this.Controls.Add(this.btnStop);
```

```
            this.Controls.Add(this.btnStart);
            this.Controls.Add(this.ctlServerAddress);
            this.Controls.Add(this.ctlServerPassword);
            this.Controls.Add(this.ctlServerLogin);
            this.Controls.Add(this.ctlServerGateway);
            this.Controls.Add(this.ctlServerAPN);
            this.Controls.Add(this.label9);
            this.Controls.Add(this.label10);
            this.Controls.Add(this.label12);
            this.Controls.Add(this.label13);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.btnCue);
            this.Controls.Add(this.buttonCapture);
            this.Controls.Add(this.label18);
            this.Controls.Add(this.label11);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.None;
            this.Menu = this.mainMenu;
            this.Name = "CaptureTest1";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Record";
            this.Load += new
System.EventHandler(this.CaptureTest_Load);

((System.ComponentModel.ISupportInitialize)(this.dS_Webcam)).EndI
nit();

((System.ComponentModel.ISupportInitialize)(this.tableNoHPBinding
Source)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

            }
            #endregion

            [STAThread]
            static void Main()
            {
                  AppDomain currentDomain =
AppDomain.CurrentDomain;
                  Application.Run(new CaptureTest1());
            }

            private void btnExit_Click(object sender,
System.EventArgs e)
            {
            //if ( capture != null )
            //    capture.Stop();
                  Application.Exit();
```

```
            }

            private void btnStart_Click(object sender,
System.EventArgs e)
            {
            timer2.Stop();
                  try
                  {
                        if ( capture == null )
                              throw new ApplicationException(
"Please select a video and/or audio device." );
                        if ( !capture.Cued )
                              capture.Filename =
txtFilename.Text;
                        capture.Start();
                //btnCue.Enabled = false;
                //btnStart.Enabled = false;
                timer1.Start();
                x = 0;
                  }
                  catch (Exception ex)
                  {
                        MessageBox.Show( ex.Message + "\n\n" +
ex.ToString() );
                  }
            }

            private void btnStop_Click(object sender,
System.EventArgs e)
            {
                  try
                  {
                //if ( capture == null )
                //    throw new ApplicationException( "Please
select a video and/or audio device." );

                        capture.Stop();
                        btnCue.Enabled = true;
                        btnStart.Enabled = true;
                timer1.Stop();
                  }
                  catch (Exception ex)
                  {
                        MessageBox.Show( ex.Message + "\n\n" +
ex.ToString() );
                  }
            }

            private void updateMenu()
            {
                  MenuItem m;
                  Filter f;
            //Source s;
            //Source current;
                  PropertyPage p;
                  Control oldPreviewWindow = null;
```

```
                    // Disable preview to avoid additional flashes
(optional)
                    if ( capture != null )
                    {
                          oldPreviewWindow = capture.PreviewWindow;
                          capture.PreviewWindow = null;
                    }

                    // Load video devices
                    Filter videoDevice = null;
                    if ( capture != null )
                          videoDevice = capture.VideoDevice;
                    mnuVideoDevices.MenuItems.Clear();
                    m = new MenuItem( "(None)", new EventHandler(
mnuVideoDevices_Click ) );
                    m.Checked = ( videoDevice == null );
                    mnuVideoDevices.MenuItems.Add( m );
                    for ( int c = 0; c <
filters.VideoInputDevices.Count; c++ )
                    {

                          f = filters.VideoInputDevices[c];
                          m = new MenuItem( f.Name, new
EventHandler( mnuVideoDevices_Click ) );
                          m.Checked = ( videoDevice == f );
                          mnuVideoDevices.MenuItems.Add( m );
                    }
                    mnuVideoDevices.Enabled = (
filters.VideoInputDevices.Count > 0 );

                    // Load audio devices
                    Filter audioDevice = null;
                    if ( capture != null )
                          audioDevice = capture.AudioDevice;
                    mnuAudioDevices.MenuItems.Clear();
                    m = new MenuItem( "(None)", new EventHandler(
mnuAudioDevices_Click ) );
                    m.Checked = ( audioDevice == null );
                    mnuAudioDevices.MenuItems.Add( m );
                    for ( int c = 0; c <
filters.AudioInputDevices.Count; c++ )
                    {
                          f = filters.AudioInputDevices[c];
                          m = new MenuItem( f.Name, new
EventHandler( mnuAudioDevices_Click ) );
                          m.Checked = ( audioDevice == f );
                          mnuAudioDevices.MenuItems.Add( m );
                    }
                    mnuAudioDevices.Enabled = (
filters.AudioInputDevices.Count > 0 );

                    // Load property pages
              try
              {
                  mnuPropertyPages.MenuItems.Clear();
                  for (int c = 0; c < capture.PropertyPages.Count;
c++)
```

```
                {
                    p = capture.PropertyPages[c];
                    m = new MenuItem( p.Name + "...", new
EventHandler( mnuPropertyPages_Click ) );
                    mnuPropertyPages.MenuItems.Add(m);
                }
                mnuPropertyPages.Enabled =
(capture.PropertyPages.Count > 0);
            }
            catch { mnuPropertyPages.Enabled = false; }

                // Check Preview menu option
                mnuPreview.Checked = ( oldPreviewWindow != null
);
                mnuPreview.Enabled = ( capture != null );

                // Reenable preview if it was enabled before
                if ( capture != null )
                    capture.PreviewWindow = oldPreviewWindow;
            }

            private void mnuVideoDevices_Click(object sender,
System.EventArgs e)
            {
                try
                {
                    // Get current devices and dispose of
capture object
                    // because the video and audio device can
only be changed
                    // by creating a new Capture object.
                    Filter videoDevice = null;
                    Filter audioDevice = null;
                    if ( capture != null )
                    {
                        videoDevice = capture.VideoDevice;
                        audioDevice = capture.AudioDevice;
                        capture.Dispose();
                        capture = null;
                    }

                    // Get new video device
                    MenuItem m = sender as MenuItem;
                    videoDevice = ( m.Index>0 ?
filters.VideoInputDevices[m.Index-1] : null );

                    // Create capture object
                    if ( ( videoDevice != null ) || (
audioDevice != null ) )
                    {
                        capture = new Capture( videoDevice,
audioDevice );
                        capture.CaptureComplete += new
EventHandler( OnCaptureComplete );
                    }

                    // Update the menu
```

```
                        updateMenu();
                //MessageBox.Show();
                  }
                  catch (Exception ex)
                  {
                        MessageBox.Show( "Video device not
supported.\n\n" + ex.Message + "\n\n" + ex.ToString() );
                  }
            }

            private void mnuAudioDevices_Click(object sender,
System.EventArgs e)
            {
                  try
                  {
                        // Get current devices and dispose of
capture object
                        // because the video and audio device can
only be changed
                        // by creating a new Capture object.
                        Filter videoDevice = null;
                        Filter audioDevice = null;
                        if ( capture != null )
                        {
                              videoDevice = capture.VideoDevice;
                              audioDevice = capture.AudioDevice;
                              capture.Dispose();
                              capture = null;
                        }

                        // Get new audio device
                        MenuItem m = sender as MenuItem;
                        audioDevice = ( m.Index>0 ?
filters.AudioInputDevices[m.Index-1] : null );

                        // Create capture object
                        if ( ( videoDevice != null ) || (
audioDevice != null ) )
                        {
                              capture = new Capture( videoDevice,
audioDevice );
                              capture.CaptureComplete += new
EventHandler( OnCaptureComplete );
                        }

                        // Update the menu
                        updateMenu();
                  }
                  catch (Exception ex)
                  {
                        MessageBox.Show( "Audio device not
supported.\n\n" + ex.Message + "\n\n" + ex.ToString() );
                  }
            }

            private void mnuExit_Click(object sender,
System.EventArgs e)
```

```
            {
                  if ( capture != null )
                        capture.Stop();
                  Application.Exit();
            }

            private void mnuPreview_Click(object sender,
System.EventArgs e)
            {
                  try
                  {
                        if ( capture.PreviewWindow == null )
                        {
                              capture.PreviewWindow = panelVideo;
                              mnuPreview.Checked = true;
                    timer2.Start();
                        }
                        else
                        {
                              capture.PreviewWindow = null;
                              mnuPreview.Checked = false;
                        }
                  }
                  catch (Exception ex)
                  {
                        MessageBox.Show( "Unable to
enable/disable preview. Please submit a bug report.\n\n" +
ex.Message + "\n\n" + ex.ToString() );
                  }
            }

        private void mnuPropertyPages_Click(object sender,
System.EventArgs e)
        {
            try
            {
                MenuItem m = sender as MenuItem;
                capture.PropertyPages[m.Index].Show(this);
                updateMenu();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Unable display property page.
Please submit a bug report.\n\n" + ex.Message + "\n\n" +
ex.ToString());
            }
        }

        private void OnCaptureComplete(object sender, EventArgs
e)
        {
            // Demonstrate the Capture.CaptureComplete event.
            Debug.WriteLine("Capture complete.");
        }

        int x = 0;
```

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            x = x + 1;
            labelTimer.Text = x.ToString();
            if (x == int.Parse(textBoxTimer.Text))
            {
                btnStop_Click(sender, e);
                mnuPreview_Click(sender, e);
                button2_Click(sender, e);
                SendMMS();
                timer1.Stop();
                //SendMMS();
                //txtReportCopy.Text = "Compressing " +
txtAviFileName.Text + " to compressed.avi...\r\n";
                //CopyFile(@"c:\compressed.mpeg", true);
                //txtReportCopy.Text += "...finished.";
            }
        }

        private void SendMMS()
        {
            MmsSlide objSlide = new MmsSlide();
            MmsMessage objMessage = new MmsMessage();
            object obj = null;

            objSlide.Clear();
            objSlide.AddText("reply");
            objSlide.AddAttachment(@"c:\test.wmv", ref obj);

            //Create MMS Message and add slide including text and
image
            objMessage.Clear();
            //objMessage.AddRecipient(objGsmIn.MessageSender,
objMmsConstants.asMMS_RECIPIENT_TO);
            objMessage.AddRecipient(messageSender,
objMmsConstants.asMMS_RECIPIENT_TO);
            objMessage.From = "Webcam Server";
            objMessage.Subject = "Video Reply";
            obj = objSlide;
            objMessage.AddSlide(ref obj);

            objMm1Protocol.Device = comboBoxHPDev.Text;
            objMm1Protocol.ProviderMMSC = ctlServerAddress.Text;
            objMm1Protocol.ProviderAPN = ctlServerAPN.Text;
            objMm1Protocol.ProviderAPNAccount =
ctlServerLogin.Text;
            objMm1Protocol.ProviderAPNPassword =
ctlServerPassword.Text;
            objMm1Protocol.ProviderWAPGateway =
ctlServerGateway.Text;
            objMm1Protocol.LogFile = textBoxLogfile.Text;

            Cursor.Current = Cursors.WaitCursor;

            // Connect
            objMm1Protocol.Connect();
            UpdateResult(objMm1Protocol.LastError);
```

```
            if (objMm1Protocol.LastError != 0)
            {
                Cursor.Current = Cursors.Default;
                return;
            }

            // Send
            obj = objMessage;
            objMm1Protocol.Send(ref obj);
            UpdateResult(objMm1Protocol.LastError);

            // Disconnect
            objMm1Protocol.Disconnect();

            Cursor.Current = Cursors.Default;
        }

        private void ReceiveSMS(string handphoneNumber, string
message, object sender, System.EventArgs e)
        {
            Cursor cur = Cursor.Current;
            Cursor.Current = Cursors.WaitCursor;

            //listView.Items.Clear();

            objGsmIn.Device = comboBoxHPDev.Text;
            objGsmIn.LogFile = textBoxLogfile.Text;

            objGsmIn.Storage = 1;


            objGsmIn.DeleteAfterReceive = 1;

            //if (comboSpeed.SelectedIndex == 0)
objGsmIn.DeviceSpeed = 0;                  // Default
            //else objGsmIn.DeviceSpeed =
System.Int32.Parse(comboSpeed.Text);

            objGsmIn.Receive();                                 //
Check for new messages

            //if (GetResult() == 0)
            //{
                //System.Windows.Forms.ListViewItem Item;

                objGsmIn.GetFirstMessage();
                int hasil = 0;
                try
                {
                    hasil =
int.Parse(table_NoHPTableAdapter.Hasil(objGsmIn.MessageSender).To
String());
                }
                catch
                {
                }
                if (hasil != 0)
```

```
                {
                    while (GetResult() == 0)
                    {
                        if
(objGsmIn.MessageData.ToString().ToUpper() == "REKAM")
                        {

                            messageSender =
objGsmIn.MessageSender;
                            btnStart_Click(sender, e);
                            //    Item =
listView.Items.Add(objGsmIn.MessageTime);

                            //
Item.SubItems.Add(objGsmIn.MessageSender);
                            //
Item.SubItems.Add(objGsmIn.MessageData);

                            //    objGsmIn.GetNextMessage();
                        }
                    }
                }
                Cursor.Current = cur;
            //}
        }

        private int GetResult()
        {
            int lResult = objGsmIn.LastError;

            textBoxResult.Text =
objGsmIn.GetErrorDescription(lResult);

            return lResult;
        }

        private void UpdateResult(System.Int32 numResult)
        {
            textBoxResult.Text = numResult.ToString() + ": " +
objMm1Protocol.GetErrorDescription(numResult);
        }

        private void buttonCapture_Click(object sender, EventArgs
e)
        {
            CaptureTest.Device d =
CaptureTest.DeviceManager.GetDevice(2);
            d.SaveImage();
        }

        private void menuItem2_Click(object sender, EventArgs e)
        {
            saveHPNumber frm = new saveHPNumber();
            frm.ShowDialog();
        }

        private void timer2_Tick(object sender, EventArgs e)
```

```
        {
            ReceiveSMS(objGsmIn.MessageSender,
objGsmIn.MessageData, sender, e);
            //timer2.Stop();
        }

        private void CaptureTest_Load(object sender, EventArgs e)
        {

this.table_NoHPTableAdapter.Fill(this.dS_Webcam.Table_NoHP);
            System.Int16 i = 0;
            for (i = 0; i < objGsmIn.GetDeviceCount(); i++)
            {

comboBoxHPDev.Items.Add(objGsmIn.GetDevice(i).ToString());
            }

            for (i = 1; i <= 8; i++)
            {
                comboBoxHPDev.Items.Add("COM" + i.ToString());
            }

            comboBoxHPDev.SelectedIndex = 0;

            if (messageSender != "")
            {
                SendMMS();
                messageSender = "";
            }
        }

#region ga kepake, bekas convert sebelumnya

        private void button1_Click(object sender, EventArgs e)
        {
            CopyFile(@"c:\compressed.mpeg", true);
        }

        private void btnCue_Click(object sender, System.EventArgs
e)
        {
            try
            {
                if (capture == null)
                    throw new ApplicationException("Please select
a video and/or audio device.");
                if (!capture.Cued)
                    capture.Filename = txtFilename.Text;
                capture.Cue();
                btnCue.Enabled = false;
                MessageBox.Show("Ready to capture.\n\nUse Cue()
before Start() to " +
                    "do all the preparation work that needs to be
done to start a " +
                    "capture. Now, when you click Start the
capture will begin faster " +
```

```
                    "than if you had just clicked Start. Using
Cue() is completely " +
                    "optional. The downside to using Cue() is the
preview is disabled until " +
                    "the capture begins.");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message + "\n\n" +
ex.ToString());
            }
        }

        private void CopyFile(String newName, bool compress)
        {
            AviManager aviManager = new
AviManager(txtFilename.Text, true);
            VideoStream aviStream = aviManager.GetVideoStream();
            VideoStream newStream;
            AviManager newManager =
aviStream.DecompressToNewFile(newName, compress, out newStream);
            aviManager.Close();
            newManager.Close();
        }


#endregion

        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            this.ShowInTaskbar = false;
            btnStop_Click(sender, e);
            convert frm = new convert();
            frm.label2.Text = txtFilename.Text;
            frm.video_index = video_index;
            frm.audio_index = audio_index;
            frm.messageSender = messageSender;
            frm.ShowDialog();
            this.Close();
        }

        private void timer3_Tick(object sender, EventArgs e)
        {
            try
            {
                capture = new
Capture(filters.VideoInputDevices[video_index],
filters.AudioInputDevices[audio_index]);
                mnuPreview_Click(sender, e);
                mnuPreview.Checked = true;
                timer3.Stop();
            }
            catch
            {
            }
        }
```

```
        }
}
```

Form : SaveHPNumber

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace CaptureTest1
{
    public partial class saveHPNumber : Form
    {
        public saveHPNumber()
        {
            InitializeComponent();
        }

        private void saveHPNumber_Load(object sender, EventArgs
e)
        {

this.table_NoHPTableAdapter1.Fill(this.dS_Webcam1.Table_NoHP);
        }

        private void buttonInsert_Click(object sender, EventArgs
e)
        {
            try
            {
                if (textBoxNoHp.Text != string.Empty)
                {

table_NoHPTableAdapter1.InsertQuery(textBoxNoHp.Text);

table_NoHPTableAdapter1.Fill(dS_Webcam1.Table_NoHP);
                }
            }
            catch
            {
                MessageBox.Show("Duplicate Data Is Not Allowed
!");
            }
        }

        private void buttonDelete_Click(object sender, EventArgs
e)
        {
            try
            {
                table_NoHPTableAdapter1.DeleteQuery(label2.Text);

table_NoHPTableAdapter1.Fill(dS_Webcam1.Table_NoHP);
```

```
            }
            catch
            {
            }
        }

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

Form : Convert

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Runtime.InteropServices;
using DirectShowLib;

namespace CaptureTest1
{
    public partial class convert : Form
    {
        public convert()
        {
            InitializeComponent();
        }
        FilterGraph fg = null;
        IGraphBuilder gb = null;
        IMediaControl mc = null;
        IMediaEventEx me = null;
        IBaseFilter matroska_mux = null;
        string fName, fExt;
        int hr;
        int WM_GRAPHNOTIFY = 0x00008001;

        public int video_index;
        public int audio_index;
        public string messageSender;

        private void button1_Click(object sender,
System.EventArgs e)
        {
            FileInfo fi = new FileInfo(label2.Text);
            fExt = fi.Extension;
            fName = fi.FullName.Substring(0, fi.FullName.Length -
fExt.Length);
            CloseInterfaces();
            InitInterfaces();
            Convert2Wmv(fName);
```

```
        }

        void InitInterfaces()
        {
            try
            {
                fg = new FilterGraph();
                gb = (IGraphBuilder)fg;
                mc = (IMediaControl)fg;
                me = (IMediaEventEx)fg;
            }
            catch (Exception)
            {
                MessageBox.Show("Couldn't start");
            }
        }

        void CloseInterfaces()
        {
            if (me != null)
            {
                hr = mc.Stop();
                DsError.ThrowExceptionForHR(hr);

                hr = me.SetNotifyWindow(IntPtr.Zero,
WM_GRAPHNOTIFY, IntPtr.Zero);
                DsError.ThrowExceptionForHR(hr);
            }
            mc = null;
            me = null;
            gb = null;
            if (matroska_mux != null)
                Marshal.ReleaseComObject(matroska_mux);
            matroska_mux = null;
            if (fg != null)
                Marshal.ReleaseComObject(fg);
            fg = null;
        }

        void Convert2Mkv(string fileName)
        {
            try
            {
                label1.Text = "processing";
                button1.Enabled = false;

                hr = me.SetNotifyWindow(this.Handle,
WM_GRAPHNOTIFY, IntPtr.Zero);
                DsError.ThrowExceptionForHR(hr);

                FileWriter file_writer = new FileWriter();

                IFileSinkFilter fs =
(IFileSinkFilter)file_writer;

                fs.SetFileName(fileName + ".mkv", null);
```

```
                DsError.ThrowExceptionForHR(hr);

                hr = gb.AddFilter((IBaseFilter)file_writer, "File
Writer");
                DsError.ThrowExceptionForHR(hr);

                Guid guid = new Guid("1E1299A2-9D42-4F12-8791-
D79E376F4143");
                Type comtype = Type.GetTypeFromCLSID(guid);
                matroska_mux =
(IBaseFilter)Activator.CreateInstance(comtype);

                hr = gb.AddFilter((IBaseFilter)matroska_mux,
"Matroska Muxer");
                DsError.ThrowExceptionForHR(hr);

                hr = gb.RenderFile(fileName + fExt, null);
                DsError.ThrowExceptionForHR(hr);

                hr = mc.Run();
                DsError.ThrowExceptionForHR(hr);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error converting to mvk: " +
ex.Message);
            }
        }

        void Convert2Wmv(string fileName)
        {
            try
            {
                label1.Text = "processing";
                button1.Enabled = false;

                hr = me.SetNotifyWindow(this.Handle,
WM_GRAPHNOTIFY, IntPtr.Zero);
                DsError.ThrowExceptionForHR(hr);

                WMAsfWriter asf_filter = new WMAsfWriter();
                IFileSinkFilter fs = (IFileSinkFilter)asf_filter;

                hr = fs.SetFileName(fileName + ".wmv", null);
                DsError.ThrowExceptionForHR(hr);

                hr = gb.AddFilter((IBaseFilter)asf_filter, "WM
Asf Writer");
                DsError.ThrowExceptionForHR(hr);

                hr = gb.RenderFile(fileName + fExt, null);
                DsError.ThrowExceptionForHR(hr);

                hr = mc.Run();
                DsError.ThrowExceptionForHR(hr);
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show("Error converting to wmv: " +
ex.Message);
            }
        }

        protected override void WndProc(ref Message m)
        {
            if (m.Msg == WM_GRAPHNOTIFY)
            {
                int p1, p2;
                EventCode code;

                if (me == null)
                    return;
                while (me.GetEvent(out code, out p1, out p2, 0)
== 0)
                {
                    if (code == EventCode.Complete)
                    {
                        label1.Text = "done";
                        button1.Enabled = true;
                        mc.Stop();
                    }

                    me.FreeEventParams(code, p1, p2);
                }
                return;
            }
            base.WndProc(ref m);
        }

        private void convert_Load(object sender, EventArgs e)
        {
            timer1.Start();
        }

        private void labelExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        int x = 0;

        private void timer1_Tick(object sender, EventArgs e)
        {
            x = x + 1;
            if (x == 10)
            {
                button1_Click(sender, e);
                timer2.Start();
            }
        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            x = x + 1;
```

```
            if (x == 30)
            {
                this.Hide();
                this.ShowInTaskbar = false;
                CaptureTest1 frm = new CaptureTest1();
                frm.video_index = video_index;
                frm.audio_index = audio_index;
                frm.messageSender = messageSender;
                frm.ShowDialog();
                this.Close();
            }
        }
    }
}
```

Class : Capture

```
using System;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Threading;
using System.Windows.Forms;
using DShowNET;


namespace DirectX.Capture
{
      public class Capture
      {
        protected enum GraphState
            {
                  Null,
                  Created,
                  Rendered,
                  Capturing
            }

            public bool Capturing { get { return(
graphState==GraphState.Capturing ); } }

            public bool Cued { get { return( isCaptureRendered &&
graphState==GraphState.Rendered ); } }

            public bool Stopped { get { return(
graphState!=GraphState.Capturing ); } }

            public string Filename
            {
                  get { return( filename ); }
                  set
                  {
                //assertStopped();
                //if (Cued)
                //{ }
```

```
                //throw new InvalidOperationException("The
Filename cannot be changed once cued. Use Stop() before changing
the filename.");
                        filename = value;
                        if ( fileWriterFilter != null )
                        {
                              string s;
                              AMMediaType mt = new AMMediaType();
                              int hr =
fileWriterFilter.GetCurFile( out s, mt );
                    if( hr < 0 ) Marshal.ThrowExceptionForHR( hr
);
                              if ( mt.formatSize > 0 )
                                    Marshal.FreeCoTaskMem(
mt.formatPtr );
                              hr = fileWriterFilter.SetFileName(
filename, mt );
                    if (hr < 0)
                    {
                        Marshal.ThrowExceptionForHR(hr);
                    }

                        }
                  }
            }

            public Control PreviewWindow
            {
                  get { return( previewWindow ); }
                  set
                  {
                //assertStopped();
                        derenderGraph();
                        previewWindow = value;
                        wantPreviewRendered = ( ( previewWindow
!= null ) && ( videoDevice != null ) );
                        renderGraph();
                        startPreviewIfNeeded();
                  }
            }

            public VideoCapabilities VideoCaps
            {
                  get
                  {
                        if ( videoCaps == null )
                        {
                              if ( videoStreamConfig != null )
                              {
                                    try
                                    {
                                          videoCaps = new
VideoCapabilities( videoStreamConfig );
                                    }
                                    catch ( Exception ex ) {
Debug.WriteLine( "VideoCaps: unable to create videoCaps." +
ex.ToString() ); }
```

```
                        }
                    }
                    return( videoCaps );
                }
            }

            public AudioCapabilities AudioCaps
            {
                get
                {
                    if ( audioCaps == null )
                    {
                        if ( audioStreamConfig != null )
                        {
                            try
                            {
                                audioCaps = new
AudioCapabilities( audioStreamConfig );
                            }
                            catch ( Exception ex ) {
Debug.WriteLine( "AudioCaps: unable to create audioCaps." +
ex.ToString() ); }
                        }
                    }
                    return( audioCaps );
                }
            }

            public Filter VideoDevice { get { return( videoDevice
); } }

            public Filter AudioDevice { get { return( audioDevice
); } }

            public Filter VideoCompressor {
                get { return( videoCompressor ); }
                set
                {
                //assertStopped();
                    destroyGraph();
                    videoCompressor = value;
                    renderGraph();
                    startPreviewIfNeeded();
                }
            }

            public Filter AudioCompressor
            {
                get { return( audioCompressor ); }
                set
                {
                //assertStopped();
                    destroyGraph();
                    audioCompressor = value;
                    renderGraph();
                    startPreviewIfNeeded();
                }
```

```
            }

            public Source VideoSource
            {
                  get { return( VideoSources.CurrentSource ); }
                  set { VideoSources.CurrentSource = value; }
            }

            public Source AudioSource
            {
                  get { return( AudioSources.CurrentSource ); }
                  set { AudioSources.CurrentSource = value; }
            }

            public SourceCollection VideoSources
            {
                  get
                  {
                        if ( videoSources == null )
                        {
                              try
                              {
                                    if ( videoDevice != null )
                                          videoSources = new
SourceCollection( captureGraphBuilder, videoDeviceFilter, true );
                                    else
                                          videoSources = new
SourceCollection();
                              }
                              catch ( Exception ex ) {
Debug.WriteLine( "VideoSources: unable to create VideoSources." +
ex.ToString() ); }
                        }
                        return ( videoSources );
                  }
            }

            public SourceCollection AudioSources
            {
                  get
                  {
                        if ( audioSources == null )
                        {
                              try
                              {
                                    if ( audioDevice != null )
                                          audioSources = new
SourceCollection( captureGraphBuilder, audioDeviceFilter, false
);
                                    else
                                          audioSources = new
SourceCollection();
                              }
                              catch ( Exception ex ) {
Debug.WriteLine( "AudioSources: unable to create AudioSources." +
ex.ToString() ); }
                        }
```

```
                        return ( audioSources );
                }
        }

        public PropertyPageCollection PropertyPages
        {
                get
                {
                        if ( propertyPages == null )
                        {
                                try
                                {
                                        propertyPages = new
PropertyPageCollection(
                                                captureGraphBuilder,
                                                videoDeviceFilter,
audioDeviceFilter,
                                                videoCompressorFilter,
audioCompressorFilter,
                                                VideoSources,
AudioSources );
                                }
                                catch ( Exception ex ) {
Debug.WriteLine( "PropertyPages: unable to get property pages." +
ex.ToString() ); }

                        }
                        return( propertyPages );
                }
        }

        public Tuner Tuner { get { return( tuner ); } }

        public double FrameRate
        {
                get
                {
                        long avgTimePerFrame = (long)
getStreamConfigSetting( videoStreamConfig, "AvgTimePerFrame" );
                        return( (double) 10000000 /
avgTimePerFrame );
                }
                set
                {
                        long avgTimePerFrame = (long) ( 10000000
/ value );
                        setStreamConfigSetting(
videoStreamConfig, "AvgTimePerFrame", avgTimePerFrame );
                }
        }

        public Size FrameSize
        {
                get
                {
                        BitmapInfoHeader bmiHeader;
```

```
                        bmiHeader = (BitmapInfoHeader)
getStreamConfigSetting( videoStreamConfig, "BmiHeader" );
                        Size size = new Size( bmiHeader.Width,
bmiHeader.Height );
                        return( size );
                  }
                  set
                  {
                        BitmapInfoHeader bmiHeader;
                        bmiHeader = (BitmapInfoHeader)
getStreamConfigSetting( videoStreamConfig, "BmiHeader" );
                        bmiHeader.Width = value.Width;
                        bmiHeader.Height = value.Height;
                        setStreamConfigSetting(
videoStreamConfig, "BmiHeader", bmiHeader );
                  }
            }

            public short AudioChannels
            {
                  get
                  {
                        short audioChannels = (short)
getStreamConfigSetting( audioStreamConfig, "nChannels" );
                        return( audioChannels );
                  }
                  set
                  {
                        setStreamConfigSetting(
audioStreamConfig, "nChannels", value );
                  }
            }

            public int AudioSamplingRate
            {
                  get
                  {
                        int samplingRate = (int)
getStreamConfigSetting( audioStreamConfig, "nSamplesPerSec" );
                        return( samplingRate );
                  }
                  set
                  {
                        setStreamConfigSetting(
audioStreamConfig, "nSamplesPerSec", value );
                  }
            }

            public short AudioSampleSize
            {
                  get
                  {
                        short sampleSize = (short)
getStreamConfigSetting( audioStreamConfig, "wBitsPerSample" );
                        return( sampleSize );
                  }
                  set
```

```
                {
                        setStreamConfigSetting(
audioStreamConfig, "wBitsPerSample", value );
                }
            }

            public event EventHandler
      CaptureComplete;

            protected GraphState          graphState =
GraphState.Null;        // State of the internal filter graph
            protected bool                        isPreviewRendered
= false;                // When graphState==Rendered, have we
rendered the preview stream?
            protected bool                        isCaptureRendered
= false;                // When graphState==Rendered, have we
rendered the capture stream?
            protected bool
      wantPreviewRendered = false;        // Do we need the
preview stream rendered (VideoDevice and PreviewWindow != null)
            protected bool
      wantCaptureRendered = false;        // Do we need the
capture stream rendered

            protected int                         rotCookie = 0;
                                    // Cookie into the Running
Object Table
            protected Filter              videoDevice = null;
                                // Property Backer: Video capture
device filter
            protected Filter              audioDevice = null;
                                // Property Backer: Audio capture
device filter
            protected Filter              videoCompressor = null;
                              // Property Backer: Video compression
filter
            protected Filter              audioCompressor = null;
                              // Property Backer: Audio compression
filter
            protected string              filename = "";
                                // Property Backer: Name of file to
capture to
            protected Control             previewWindow = null;
                              // Property Backer: Owner control for
preview
            protected VideoCapabilities   videoCaps = null;
                              // Property Backer: capabilities of video
device
            protected AudioCapabilities   audioCaps = null;
                              // Property Backer: capabilities of audio
device
            protected SourceCollection    videoSources = null;
                              // Property Backer: list of physical
video sources
            protected SourceCollection    audioSources = null;
                              // Property Backer: list of physical
audio sources
```

```
            protected PropertyPageCollection propertyPages =
null;             // Property Backer: list of property pages
exposed by filters
            protected Tuner                     tuner = null;
                                    // Property Backer: TV Tuner

            protected IGraphBuilder       graphBuilder;
                                // DShow Filter: Graph builder
            protected IMediaControl       mediaControl;
                                // DShow Filter: Start/Stop the
filter graph -> copy of graphBuilder
            protected IVideoWindow        videoWindow;
                                // DShow Filter: Control preview
window -> copy of graphBuilder
            protected ICaptureGraphBuilder2
      captureGraphBuilder = null;    // DShow Filter: building
graphs for capturing video
            protected IAMStreamConfig     videoStreamConfig =
null;             // DShow Filter: configure frame rate, size
            protected IAMStreamConfig     audioStreamConfig =
null;             // DShow Filter: configure sample rate, sample
size
            protected IBaseFilter         videoDeviceFilter =
null;             // DShow Filter: selected video device
            protected IBaseFilter         videoCompressorFilter =
null;       // DShow Filter: selected video compressor
            protected IBaseFilter         audioDeviceFilter =
null;             // DShow Filter: selected audio device
            protected IBaseFilter         audioCompressorFilter =
null;       // DShow Filter: selected audio compressor
            protected IBaseFilter         muxFilter = null;
                        // DShow Filter: multiplexor (combine
video and audio streams)
            protected IFileSinkFilter     fileWriterFilter =
null;             // DShow Filter: file writer

            public Capture(Filter videoDevice, Filter
audioDevice)
            {
                  if ( videoDevice == null && audioDevice == null
)
                        throw new ArgumentException( "The
videoDevice and/or the audioDevice parameter must be set to a
valid Filter.\n" );
                  this.videoDevice = videoDevice;
                  this.audioDevice = audioDevice;
                  this.Filename = getTempFilename();
                  createGraph();
            }

            ~Capture()
            {
                  Dispose();
            }

        public void Cue()
            {
```

```
            //assertStopped();

                  wantCaptureRendered = true;

                  renderGraph();

                  int hr = mediaControl.Pause();
                  if ( hr != 0 ) Marshal.ThrowExceptionForHR( hr
);
            }

            public void Start()
            {
            //assertStopped();

                  wantCaptureRendered = true;

                  renderGraph();

                  int hr = mediaControl.Run();
            if (hr != 0) Marshal.ThrowExceptionForHR(hr);

                  graphState = GraphState.Capturing;
            }

            public void Stop()
            {
                  wantCaptureRendered = false;

                  if ( mediaControl != null )
                  {
                        mediaControl.Stop();
                  }

                  if ( graphState == GraphState.Capturing )
                  {
                        graphState = GraphState.Rendered;
                        if ( CaptureComplete != null )
                              CaptureComplete( this, null );
                  }

                  try { renderGraph(); } catch {}
                  try { startPreviewIfNeeded(); } catch {}
            }

            public void Dispose()
            {
                  wantPreviewRendered = false;
                  wantCaptureRendered = false;
                  CaptureComplete = null;

                  try { destroyGraph(); } catch {}

                  if ( videoSources != null )
                        videoSources.Dispose(); videoSources =
null;
                  if ( audioSources != null )
```

```
                        audioSources.Dispose(); audioSources =
null;
            }

        protected void createGraph()
            {
                  Guid                          cat;
                  Guid                          med;
                  int                                 hr;

                  if ( videoDevice == null && audioDevice == null
)
                        throw new ArgumentException( "The video
and/or audio device have not been set. Please set one or both to
valid capture devices.\n" );

                  if ( (int)graphState < (int)GraphState.Created
)
                  {
                        GC.Collect();

                        graphBuilder = (IGraphBuilder)
Activator.CreateInstance( Type.GetTypeFromCLSID(
Clsid.FilterGraph, true ) );

                        Guid clsid = Clsid.CaptureGraphBuilder2;
                        Guid riid =
typeof(ICaptureGraphBuilder2).GUID;
                        captureGraphBuilder =
(ICaptureGraphBuilder2) DsBugWO.CreateDsInstance( ref clsid, ref
riid );

                        hr = captureGraphBuilder.SetFiltergraph(
graphBuilder );
                        if( hr < 0 ) Marshal.ThrowExceptionForHR(
hr );

                        #if DEBUG
                        DsROT.AddGraphToRot( graphBuilder, out
rotCookie );
                        #endif

                        if ( VideoDevice != null )
                        {
                              videoDeviceFilter = (IBaseFilter)
Marshal.BindToMoniker( VideoDevice.MonikerString );
                              hr = graphBuilder.AddFilter(
videoDeviceFilter, "Video Capture Device" );
                              if( hr < 0 )
Marshal.ThrowExceptionForHR( hr );
                        }

                        if ( AudioDevice != null )
                        {
                              audioDeviceFilter = (IBaseFilter)
Marshal.BindToMoniker( AudioDevice.MonikerString );
```

```
                              hr = graphBuilder.AddFilter(
audioDeviceFilter, "Audio Capture Device" );
                              if( hr < 0 )
Marshal.ThrowExceptionForHR( hr );
                        }

                        if ( VideoCompressor != null )
                        {
                              videoCompressorFilter =
(IBaseFilter) Marshal.BindToMoniker(
VideoCompressor.MonikerString );
                              hr = graphBuilder.AddFilter(
videoCompressorFilter, "Video Compressor" );
                              if( hr < 0 )
Marshal.ThrowExceptionForHR( hr );
                        }

                        if ( AudioCompressor != null )
                        {
                              audioCompressorFilter =
(IBaseFilter) Marshal.BindToMoniker(
AudioCompressor.MonikerString );
                              hr = graphBuilder.AddFilter(
audioCompressorFilter, "Audio Compressor" );
                              if( hr < 0 )
Marshal.ThrowExceptionForHR( hr );
                        }

                        object o;
                        cat = PinCategory.Capture;
                        med = MediaType.Interleaved;
                        Guid iid = typeof(IAMStreamConfig).GUID;
                        hr = captureGraphBuilder.FindInterface(
                              ref cat, ref med,
videoDeviceFilter, ref iid, out o );

                        if ( hr != 0 )
                        {
                              med = MediaType.Video;
                              hr =
captureGraphBuilder.FindInterface(
                                    ref cat, ref med,
videoDeviceFilter, ref iid, out o );

                              if ( hr != 0 )
                                    o = null;
                        }
                        videoStreamConfig = o as IAMStreamConfig;

                        o = null;
                        cat = PinCategory.Capture;
                        med = MediaType.Audio ;
                        iid = typeof(IAMStreamConfig).GUID;
                        hr = captureGraphBuilder.FindInterface(
                              ref cat, ref med,
audioDeviceFilter, ref iid, out o );
                        if ( hr != 0 )
```

```
                              o = null;
                        audioStreamConfig = o as IAMStreamConfig;

                        mediaControl = (IMediaControl)
graphBuilder;

                        if ( videoSources != null )
videoSources.Dispose(); videoSources = null;

                        if ( audioSources != null )
audioSources.Dispose(); audioSources = null;

                        if ( propertyPages != null )
propertyPages.Dispose(); propertyPages = null;

                        videoCaps = null;

                        audioCaps = null;

                        o = null;
                        cat = PinCategory.Capture;
                        med = MediaType.Interleaved;
                        iid = typeof(IAMTVTuner).GUID;
                        hr = captureGraphBuilder.FindInterface(
                              ref cat, ref med,
videoDeviceFilter, ref iid, out o );
                        if ( hr != 0 )
                        {
                              med = MediaType.Video ;
                              hr =
captureGraphBuilder.FindInterface(
                                    ref cat, ref med,
videoDeviceFilter, ref iid, out o );
                              if ( hr != 0 )
                                    o = null;
                        }
                        IAMTVTuner t = o as IAMTVTuner;
                        if ( t != null )
                              tuner = new Tuner( t );

                        graphState = GraphState.Created;
                  }
            }

            protected void renderGraph()
            {
                  Guid                        cat;
                  Guid                        med;
                  int                               hr;
                  bool                        didSomething =
false;
                  const int WS_CHILD                =
0x40000000;
                  const int WS_CLIPCHILDREN     = 0x02000000;
                  const int WS_CLIPSIBLINGS     = 0x04000000;

            //assertStopped();
```

```
                  if ( filename == null )
                        throw new ArgumentException( "The
Filename property has not been set to a file.\n" );

                  if ( mediaControl != null )
                        mediaControl.Stop();

                  createGraph();

                  if ( !wantPreviewRendered && isPreviewRendered
)
                        derenderGraph();
                  if ( !wantCaptureRendered && isCaptureRendered
)
                        if ( wantPreviewRendered )
                              derenderGraph();


                  if ( wantCaptureRendered && !isCaptureRendered
)
                  {
                        Guid mediaSubType = MediaSubType.Avi;
                        hr =
captureGraphBuilder.SetOutputFileName( ref mediaSubType,
Filename, out muxFilter, out fileWriterFilter );
                        if( hr < 0 ) Marshal.ThrowExceptionForHR(
hr );

                        if ( VideoDevice != null )
                        {
                              cat = PinCategory.Capture;
                              med = MediaType.Interleaved;
                              hr =
captureGraphBuilder.RenderStream( ref cat, ref med,
videoDeviceFilter, videoCompressorFilter, muxFilter );
                              if( hr < 0 )
                              {
                                    med = MediaType.Video;
                                    hr =
captureGraphBuilder.RenderStream( ref cat, ref med,
videoDeviceFilter, videoCompressorFilter, muxFilter );
                                    if ( hr == -2147220969 )
throw new DeviceInUseException( "Video device", hr );
                                    if( hr < 0 )
Marshal.ThrowExceptionForHR( hr );
                              }
                        }

                        if ( AudioDevice != null )
                        {
                              cat = PinCategory.Capture;
                              med = MediaType.Audio;
                              hr =
captureGraphBuilder.RenderStream( ref cat, ref med,
audioDeviceFilter, audioCompressorFilter, muxFilter );
```

```
                              if( hr < 0 )
Marshal.ThrowExceptionForHR( hr );
                        }

                        isCaptureRendered = true;
                        didSomething = true;
                  }

                  if ( wantPreviewRendered && !isPreviewRendered
)
                  {
                        cat = PinCategory.Preview;
                        med = MediaType.Video;
                        hr = captureGraphBuilder.RenderStream(
ref cat, ref med, videoDeviceFilter, null, null );
                        if( hr < 0 ) Marshal.ThrowExceptionForHR(
hr );

                        videoWindow = (IVideoWindow)
graphBuilder;

                        hr = videoWindow.put_Owner(
previewWindow.Handle );
                        if( hr < 0 ) Marshal.ThrowExceptionForHR(
hr );

                        hr = videoWindow.put_WindowStyle(
WS_CHILD | WS_CLIPCHILDREN | WS_CLIPSIBLINGS);
                        if( hr < 0 ) Marshal.ThrowExceptionForHR(
hr );

                        previewWindow.Resize += new EventHandler(
onPreviewWindowResize );
                        onPreviewWindowResize( this, null );

                        hr = videoWindow.put_Visible(
DsHlp.OATRUE );
                        if( hr < 0 ) Marshal.ThrowExceptionForHR(
hr );

                        isPreviewRendered = true;
                        didSomething = true;
                  }

                  if ( didSomething )
                        graphState = GraphState.Rendered;
            }

            protected void startPreviewIfNeeded()
            {
                  if ( wantPreviewRendered && isPreviewRendered
&& !isCaptureRendered )
                  {
                        mediaControl.Run();
                  }
            }
```

```
            protected void derenderGraph()
            {
                  if ( mediaControl != null )
                        mediaControl.Stop();

                  if ( videoWindow != null )
                  {
                        videoWindow.put_Visible( DsHlp.OAFALSE );
                        videoWindow.put_Owner( IntPtr.Zero );
                        videoWindow = null;
                  }

                  if ( PreviewWindow != null )
                        previewWindow.Resize -= new EventHandler(
onPreviewWindowResize );

                  if ( (int)graphState >=
(int)GraphState.Rendered )
                  {
                        graphState = GraphState.Created;
                        isCaptureRendered = false;
                        isPreviewRendered = false;

                        if ( videoDeviceFilter != null )
                              removeDownstream(
videoDeviceFilter, (videoCompressor==null) );
                        if ( audioDeviceFilter != null )
                              removeDownstream(
audioDeviceFilter, (audioCompressor==null) );

                        muxFilter = null;
                        fileWriterFilter = null;
                  }
            }

            protected void removeDownstream( IBaseFilter filter,
bool removeFirstFilter )
            {
                  IEnumPins pinEnum;
                  int hr = filter.EnumPins( out pinEnum );
                  pinEnum.Reset();
                  if( (hr == 0) && (pinEnum != null) )
                  {
                        IPin[] pins = new IPin[1];
                        int f;
                        do
                        {
                              hr = pinEnum.Next( 1, pins, out f
);
                              if( (hr == 0) && (pins[0] != null)
)
                              {
                                    IPin pinTo = null;
                                    pins[0].ConnectedTo( out
pinTo );
                                    if ( pinTo != null )
                                    {
```

```
                                                PinInfo info = new
PinInfo();
                                                hr =
pinTo.QueryPinInfo( out info );
                                                if( (hr == 0) &&
(info.dir == (PinDirection.Input)) )
                                                {
                                                      removeDownstream(
info.filter, true );

      graphBuilder.Disconnect( pinTo );

      graphBuilder.Disconnect( pins[0] );

                                                      if ( (
info.filter != videoCompressorFilter ) &&
                                                            (
info.filter != audioCompressorFilter ) )

      graphBuilder.RemoveFilter( info.filter );
                                                }

      Marshal.ReleaseComObject( info.filter );

      Marshal.ReleaseComObject( pinTo );
                                          }
                                          Marshal.ReleaseComObject(
pins[0] );
                                    }
                              }
                              while( hr == 0 );

                              Marshal.ReleaseComObject( pinEnum );
pinEnum = null;
                        }
                  }

                  protected void destroyGraph()
                  {
                        try{ derenderGraph(); } catch {}

                        graphState = GraphState.Null;
                        isCaptureRendered = false;
                        isPreviewRendered = false;

                        if ( rotCookie != 0 )
                        {
                              DsROT.RemoveGraphFromRot( ref rotCookie
);
                              rotCookie = 0;
                        }

                        if ( muxFilter != null )
                              graphBuilder.RemoveFilter( muxFilter );
                        if ( videoCompressorFilter != null )
```

```
                        graphBuilder.RemoveFilter(
videoCompressorFilter  );
                if ( audioCompressorFilter != null )
                        graphBuilder.RemoveFilter(
audioCompressorFilter  );
                if ( videoDeviceFilter != null )
                        graphBuilder.RemoveFilter(
videoDeviceFilter );
                if ( audioDeviceFilter != null )
                        graphBuilder.RemoveFilter(
audioDeviceFilter );

                if ( videoSources != null )
                        videoSources.Dispose(); videoSources =
null;
                if ( audioSources != null )
                        audioSources.Dispose(); audioSources =
null;
                if ( propertyPages != null )
                        propertyPages.Dispose(); propertyPages =
null;
                if ( tuner != null )
                        tuner.Dispose(); tuner = null;

                if ( graphBuilder != null )
                        Marshal.ReleaseComObject( graphBuilder );
graphBuilder = null;
                if ( captureGraphBuilder != null )
                        Marshal.ReleaseComObject(
captureGraphBuilder ); captureGraphBuilder = null;
                if ( muxFilter != null )
                        Marshal.ReleaseComObject( muxFilter );
muxFilter = null;
                if ( fileWriterFilter != null )
                        Marshal.ReleaseComObject(
fileWriterFilter ); fileWriterFilter = null;
                if ( videoDeviceFilter != null )
                        Marshal.ReleaseComObject(
videoDeviceFilter ); videoDeviceFilter = null;
                if ( audioDeviceFilter != null )
                        Marshal.ReleaseComObject(
audioDeviceFilter ); audioDeviceFilter = null;
                if ( videoCompressorFilter != null )
                        Marshal.ReleaseComObject(
videoCompressorFilter ); videoCompressorFilter = null;
                if ( audioCompressorFilter != null )
                        Marshal.ReleaseComObject(
audioCompressorFilter ); audioCompressorFilter = null;

                mediaControl = null;
                videoWindow = null;

                GC.Collect();
            }

            protected void onPreviewWindowResize(object sender,
EventArgs e)
```

```
            {
                  if ( videoWindow != null )
                  {
                        Rectangle rc =
previewWindow.ClientRectangle;
                        videoWindow.SetWindowPosition( 0, 0,
rc.Right, rc.Bottom );
                  }
            }

            protected string getTempFilename()
            {
                  string s;
                  try
                  {
                        int count = 0;
                        int i;
                        Random r = new Random();
                        string tempPath = Path.GetTempPath();
                        do
                        {
                              i = r.Next();
                              s = Path.Combine( tempPath,
i.ToString("X") + ".avi" );
                              count++;
                              if ( count > 100 ) throw new
InvalidOperationException( "Unable to find temporary file." );
                        } while ( File.Exists( s ) );
                  }
                  catch { s = "c:\temp.avi"; }
                  return( s );
            }

            protected object getStreamConfigSetting(
IAMStreamConfig streamConfig, string fieldName)
            {
                  if ( streamConfig == null )
                        throw new NotSupportedException();
            //assertStopped();
                  derenderGraph();

                  object returnValue = null;
                  IntPtr pmt = IntPtr.Zero;
                  AMMediaType mediaType = new AMMediaType();

                  try
                  {
                        int hr = streamConfig.GetFormat( out pmt
);
                        if ( hr != 0 )
                              Marshal.ThrowExceptionForHR( hr );
                        Marshal.PtrToStructure( pmt, mediaType );

                        object formatStruct;
                        if ( mediaType.formatType ==
FormatType.WaveEx )
                              formatStruct = new WaveFormatEx();
```

```
                        else if ( mediaType.formatType ==
FormatType.VideoInfo )
                              formatStruct = new
VideoInfoHeader();
                        else if ( mediaType.formatType ==
FormatType.VideoInfo2 )
                              formatStruct = new
VideoInfoHeader2();
                        else
                              throw new NotSupportedException(
"This device does not support a recognized format block." );

                        Marshal.PtrToStructure(
mediaType.formatPtr, formatStruct );

                        Type structType = formatStruct.GetType();
                        FieldInfo fieldInfo =
structType.GetField( fieldName );
                        if ( fieldInfo == null )
                              throw new NotSupportedException(
"Unable to find the member '" + fieldName + "' in the format
block." );

                        returnValue = fieldInfo.GetValue(
formatStruct );

                  }
                  finally
                  {
                        DsUtils.FreeAMMediaType( mediaType );
                        Marshal.FreeCoTaskMem( pmt );
                  }
                  renderGraph();
                  startPreviewIfNeeded();

                  return( returnValue );
            }

            protected object setStreamConfigSetting(
IAMStreamConfig streamConfig, string fieldName, object newValue)
            {
                  if ( streamConfig == null )
                        throw new NotSupportedException();
            //assertStopped();
                  derenderGraph();

                  object returnValue = null;
                  IntPtr pmt = IntPtr.Zero;
                  AMMediaType mediaType = new AMMediaType();

                  try
                  {
                        int hr = streamConfig.GetFormat( out pmt
);
                        if ( hr != 0 )
                              Marshal.ThrowExceptionForHR( hr );
                        Marshal.PtrToStructure( pmt, mediaType );
```

```
                        object formatStruct;
                        if ( mediaType.formatType ==
FormatType.WaveEx )
                              formatStruct = new WaveFormatEx();
                        else if ( mediaType.formatType ==
FormatType.VideoInfo )
                              formatStruct = new
VideoInfoHeader();
                        else if ( mediaType.formatType ==
FormatType.VideoInfo2 )
                              formatStruct = new
VideoInfoHeader2();
                        else
                              throw new NotSupportedException(
"This device does not support a recognized format block." );

                        Marshal.PtrToStructure(
mediaType.formatPtr, formatStruct );

                        Type structType = formatStruct.GetType();
                        FieldInfo fieldInfo =
structType.GetField( fieldName );
                        if ( fieldInfo == null )
                              throw new NotSupportedException(
"Unable to find the member '" + fieldName + "' in the format
block." );

                        fieldInfo.SetValue( formatStruct,
newValue );

                        Marshal.StructureToPtr( formatStruct,
mediaType.formatPtr, false );

                        hr = streamConfig.SetFormat( mediaType );
                        if ( hr != 0 )
                              Marshal.ThrowExceptionForHR( hr );
                  }
                  finally
                  {
                        DsUtils.FreeAMMediaType( mediaType );
                        Marshal.FreeCoTaskMem( pmt );
                  }
                  renderGraph();
                  startPreviewIfNeeded();

                  return( returnValue );
            }

            protected void assertStopped()
            {
            if (!Stopped)
            { }
            //    throw new InvalidOperationException( "This
operation not allowed while Capturing. Please Stop the current
capture." );
            }
```

```
        }
}
```

Class : DeviceManager

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Collections;
using System.Runtime.InteropServices;

namespace CaptureTest
{
    public class DeviceManager
    {
        [DllImport("avicap32.dll")]
        protected static extern bool
capGetDriverDescriptionA(short wDriverIndex,
            [MarshalAs(UnmanagedType.VBByRefStr)]ref String
lpszName,
           int cbName, [MarshalAs(UnmanagedType.VBByRefStr)] ref
String lpszVer, int cbVer);

        static ArrayList devices = new ArrayList();

        public static Device[] GetAllDevices()
        {
            String dName = "".PadRight(100);
            String dVersion = "".PadRight(100);

            for (short i = 0; i < 10; i++)
            {
                if (capGetDriverDescriptionA(i, ref dName, 100,
ref dVersion, 100))
                {
                    Device d = new Device(i);
                    d.Name = dName.Trim();
                    d.Version = dVersion.Trim();

                    devices.Add(d);
                }
            }

            return (Device[])devices.ToArray(typeof(Device));
        }

        public static Device GetDevice(int deviceIndex)
        {
            return (Device)devices[deviceIndex];
        }
    }
}
```

Class : Device

```
using System;
using System.Collections.Generic;
```

```
using System.Text;
using System.Runtime.InteropServices;
using System.Drawing;
using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace CaptureTest
{
    public class Device
    {
        private const short WM_CAP = 0x400;
        private const int WM_CAP_DRIVER_CONNECT = 0x40a;
        private const int WM_CAP_DRIVER_DISCONNECT = 0x40b;
        private const int WM_CAP_EDIT_COPY = 0x41e;
        private const int WM_CAP_SET_PREVIEW = 0x432;
        private const int WM_CAP_SET_OVERLAY = 0x433;
        private const int WM_CAP_SET_PREVIEWRATE = 0x434;
        private const int WM_CAP_SET_SCALE = 0x435;
        private const int WS_CHILD = 0x40000000;
        private const int WS_VISIBLE = 0x10000000;
        private const short SWP_NOMOVE = 0x2;
        //private short SWP_NOZORDER = 0x4;
        //private short HWND_BOTTOM = 1;
        //int hwnd;

        [DllImport("avicap32.dll")]
        protected static extern int
capCreateCaptureWindowA([MarshalAs(UnmanagedType.VBByRefStr)] ref
string lpszWindowName,
            int dwStyle, int x, int y, int nWidth, int nHeight,
int hWndParent, int nID);

        [DllImport("user32", EntryPoint = "SendMessageA")]
        protected static extern int SendMessage(int hwnd, int
wMsg, int wParam, [MarshalAs(UnmanagedType.AsAny)] object
lParam);

        [DllImport("user32")]
        protected static extern int SetWindowPos(int hwnd, int
hWndInsertAfter, int x, int y, int cx, int cy, int wFlags);

        [DllImport("user32")]
        protected static extern bool DestroyWindow(int hwnd);

        int index;
        int deviceHandle;
        int DeviceID = 0;
        int hHwnd = 0;

        private PictureBox container;

        public PictureBox Container
        {
            get { return container; }
            set { container = value; }
        }
```

```
        public Device(int index)
        {
            this.index = index;
        }

        private string _name;

        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }

        private string _version;

        public string Version
        {
            get { return _version; }
            set { _version = value; }
        }

        public override string ToString()
        {
            return this.Name;
        }

        public void Init(int windowHeight, int windowWidth, int
handle)
        {
            string deviceIndex = Convert.ToString(this.index);
            deviceHandle = capCreateCaptureWindowA(ref
deviceIndex, WS_VISIBLE | WS_CHILD, 0, 0, windowWidth,
windowHeight, handle, 0);

            if (SendMessage(deviceHandle, WM_CAP_DRIVER_CONNECT,
this.index, 0) > 0)
            {
                SendMessage(deviceHandle, WM_CAP_SET_SCALE, -1,
0);
                SendMessage(deviceHandle, WM_CAP_SET_PREVIEWRATE,
0x42, 0);
                SendMessage(deviceHandle, WM_CAP_SET_PREVIEW, -1,
0);

                SetWindowPos(deviceHandle, 1, 0, 0, windowWidth,
windowHeight, 6);
            }
        }

        public void
ShowWindow(global::System.Windows.Forms.Control windowsControl)
        {
            Init(windowsControl.Height, windowsControl.Width,
windowsControl.Handle.ToInt32());
        }
```

```
        public void Stop()
        {
            SendMessage(deviceHandle, WM_CAP_DRIVER_DISCONNECT,
this.index, 0);

            DestroyWindow(deviceHandle);
        }

        void CloseConnection()
        {

            SendMessage(hHwnd, WM_CAP_DRIVER_DISCONNECT,
DeviceID, 0);
            DestroyWindow(hHwnd);
        }

        public void SaveImage()
        {
            IDataObject data;
            Image oImage;
            SaveFileDialog sfdImage = new SaveFileDialog();
            sfdImage.Filter = "(*.bmp)|*.bmp";
            SendMessage(hHwnd, WM_CAP_EDIT_COPY, 0, 0);

            data = Clipboard.GetDataObject();
            if
(data.GetDataPresent(typeof(System.Drawing.Bitmap)))
            {
                oImage =
(Image)data.GetData(typeof(System.Drawing.Bitmap));
                Container.Image = oImage;
                CloseConnection();
                if (sfdImage.ShowDialog() == DialogResult.OK)
                {
                    oImage.Save(sfdImage.FileName,
System.Drawing.Imaging.ImageFormat.Bmp);
                }
            }
        }
    }
}
```

Class : Filters

```
using System;
using DShowNET;

namespace DirectX.Capture
{
    public class Filters
    {
        public FilterCollection VideoInputDevices = new
FilterCollection(FilterCategory.VideoInputDevice);

        public FilterCollection AudioInputDevices = new
FilterCollection(FilterCategory.AudioInputDevice);
```

```
        public FilterCollection VideoCompressors = new
FilterCollection(FilterCategory.VideoCompressorCategory);

        public FilterCollection AudioCompressors = new
FilterCollection(FilterCategory.AudioCompressorCategory);
    }
}
```