Savoy ActiveX Control
User Guide

Jazz Soft, Inc.

1 Revision History

Version	Date	Name	Description
1.00	Jul, 31st, 2009	Hikaru Okada	Created as new document
1.00a	Aug, 22nd, 2009	Hikaru Okada	Splitted into separate document, since number of pages became large.
1.00b	Oct, 23rd, 2009	Hikaru Okada	Corrected misspelling
1.00c	Dec, 25th, 2009	Hikaru Okada	Added procedure for Windows Vista and Windows 7.
1.00d	Dec, 18th, 2010	Hikaru Okada	Added product selection during installation. Modified for Windows 64-bit edition.
1.00e	Jul, 13th, 2014	Carl Hikaru Okada	Added description about Windows 8.1

2 Table of Contents

3 System Requirement

- Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8 or Windows 8.1.
- ActiveX compliant development environment such as Visual Basic and Visual C++.

4 Setup

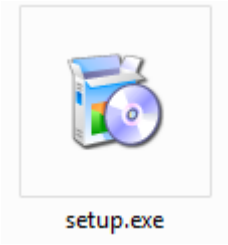
Please login Windows with Administrator privilege. If User Account Control (UAC) was enabled in Windows Vista, Windows 7, Windows 8 or Windows 8.1, installation might fail. Please temporarily turn off the UAC.

4.1Install Jazz Soft Semiconductor Solution

Setting up development environment and runtime environment are same. Run setup.exe to install.

1 If previous version of Jazz Soft Semiconductor Solution was installed on the computer, please uninstall it first. Uninstallation may be available in “Programs and Features” from Control Panel.

2 Launch setup.exe. If Microsoft Visual C++ 2008 SP1 redistributable files were not installed, they would be installed first. Also .NET Framework 3.5 would be required to run setup.



3 The setup wizard for Jazz Soft Semiconductor Solution will appear on the screen. Click “Next” button.

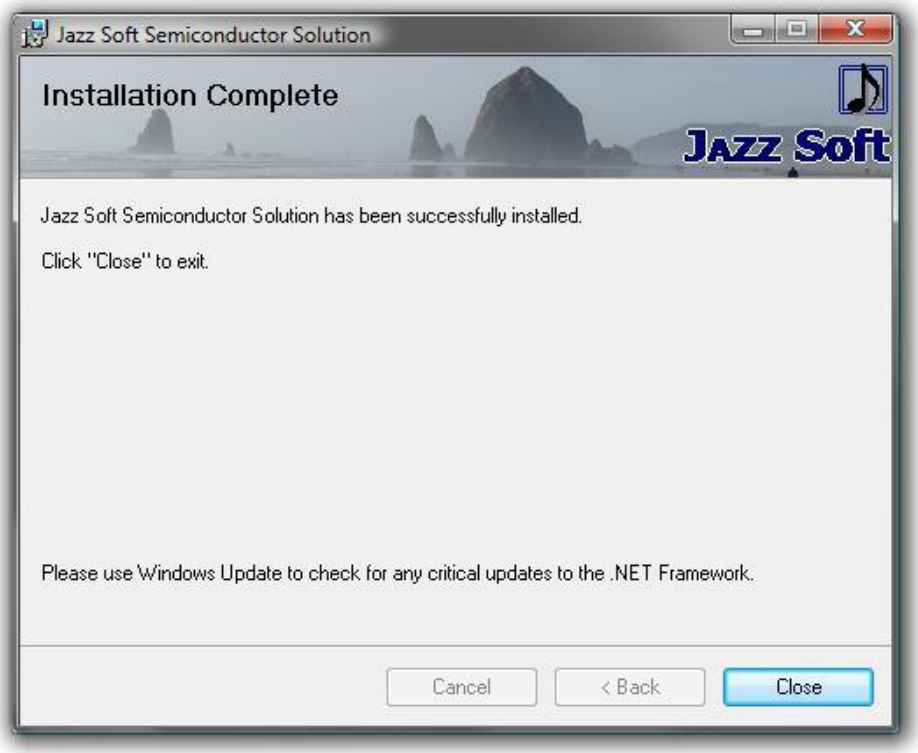
4 Select installation folder and user account. Usually, don't change anything and click "Next" button.

5 Product selection screen will appear. Click "Next" button.

6 Confirmation screen will appear. Click “Next” button.

7 Actuall installation will start.

8 Once installation was completed successfully, following screen will appear. Click “Close” button.

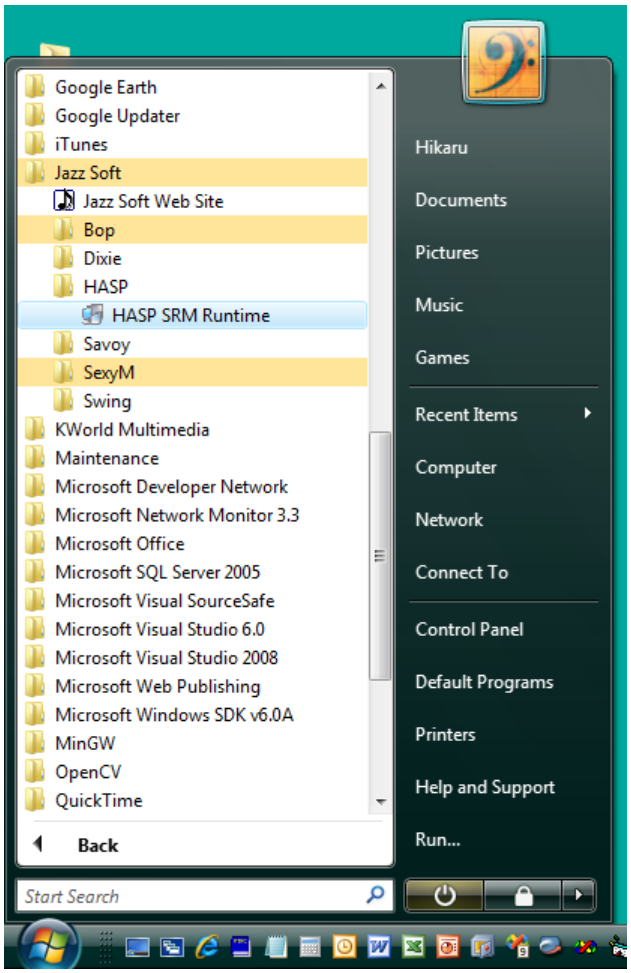


4.2Install HASP Driver

HASP key driver is not necessary for evaluation version, but is necessary to run as retail version. Required software to install HASP driver was copied by installer of Jazz Soft Semiconductor Solution

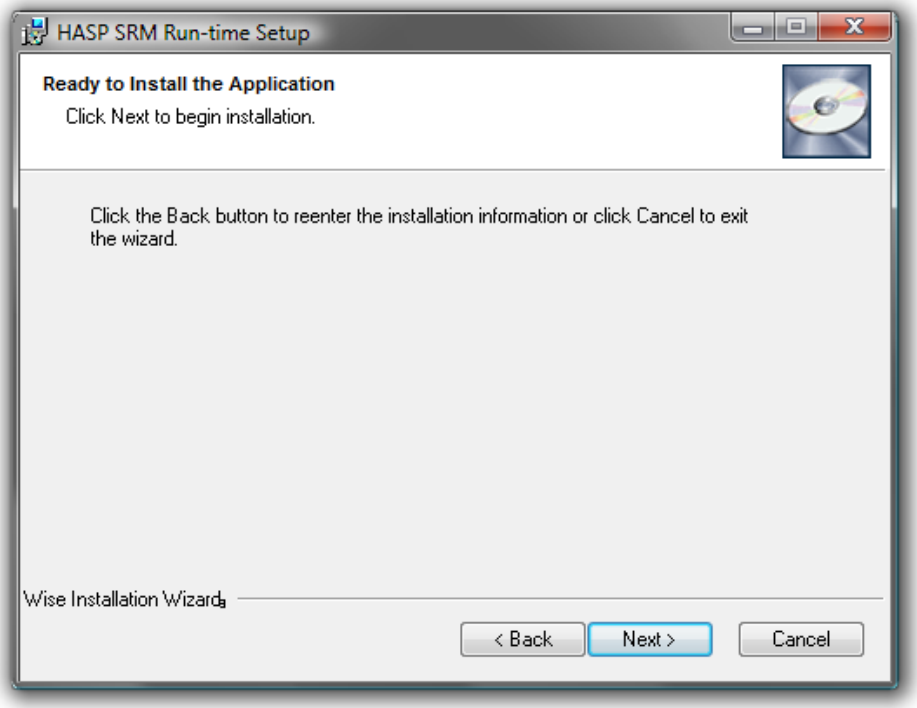
Please don't insert HASP key when installing HASP driver.

1 Click "HASP SRM Runtime" icon in [Jazz Soft] – [HASP] from Start menu.

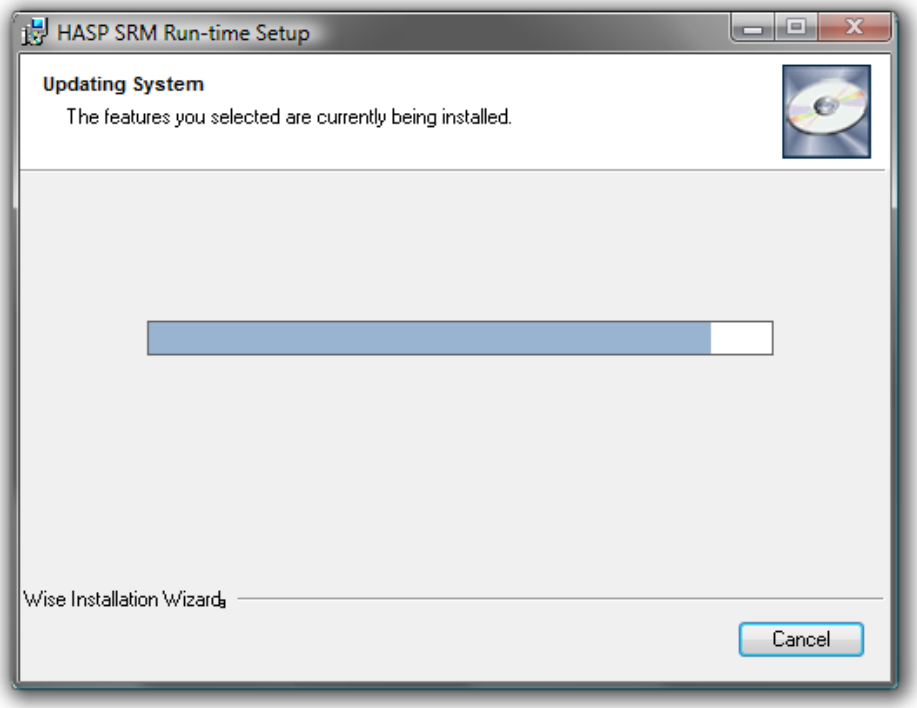


2 Welcome screen will appear. Click “Next” button.

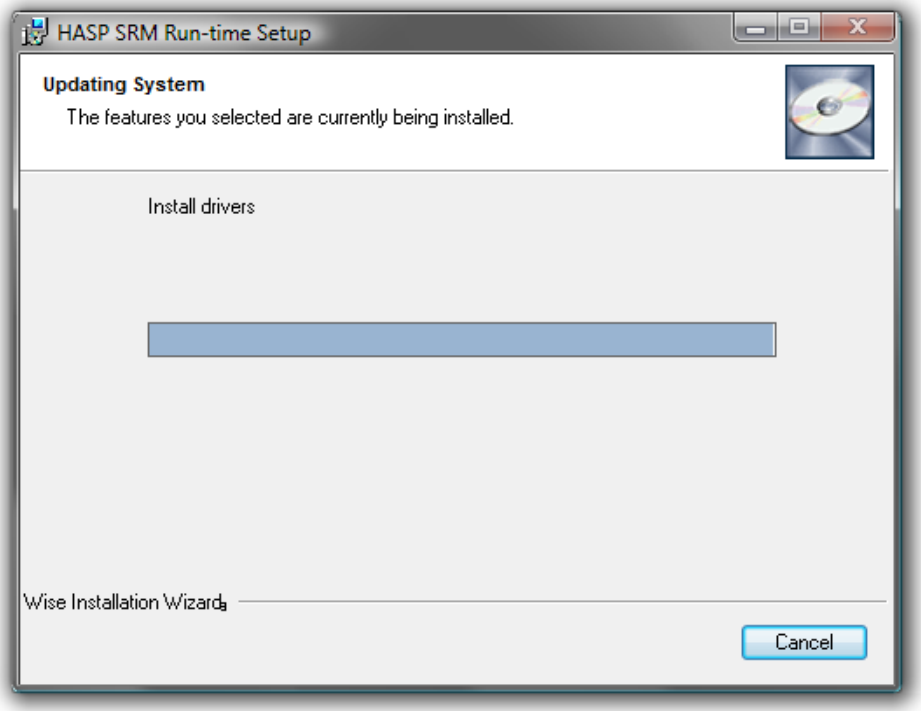
3 Confirmation screen will appear. Click “Install” button.



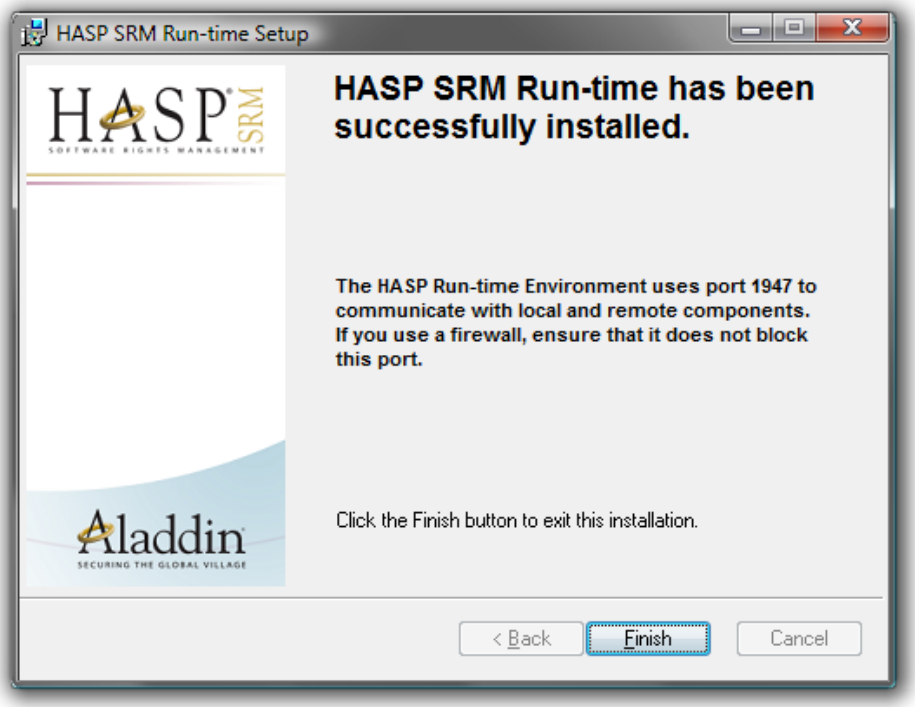
4 Actual installation will start.



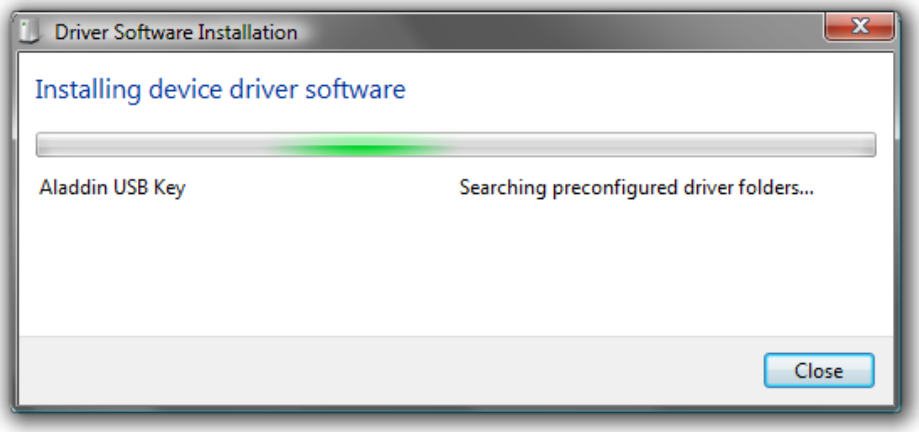
5 Text on the screen will change to “Install drivers”. It may take several minutes.



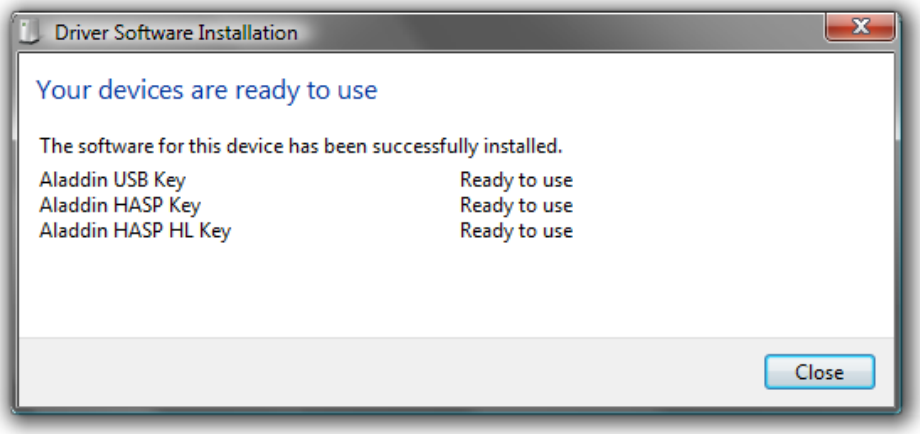
6 Once driver files are copied into computer successfully, following screen will appear. Click “Finish” button.



7 Insert HASP key in computer. Following popup dialog box will appear at lower right corner of screen on Windows Vista.



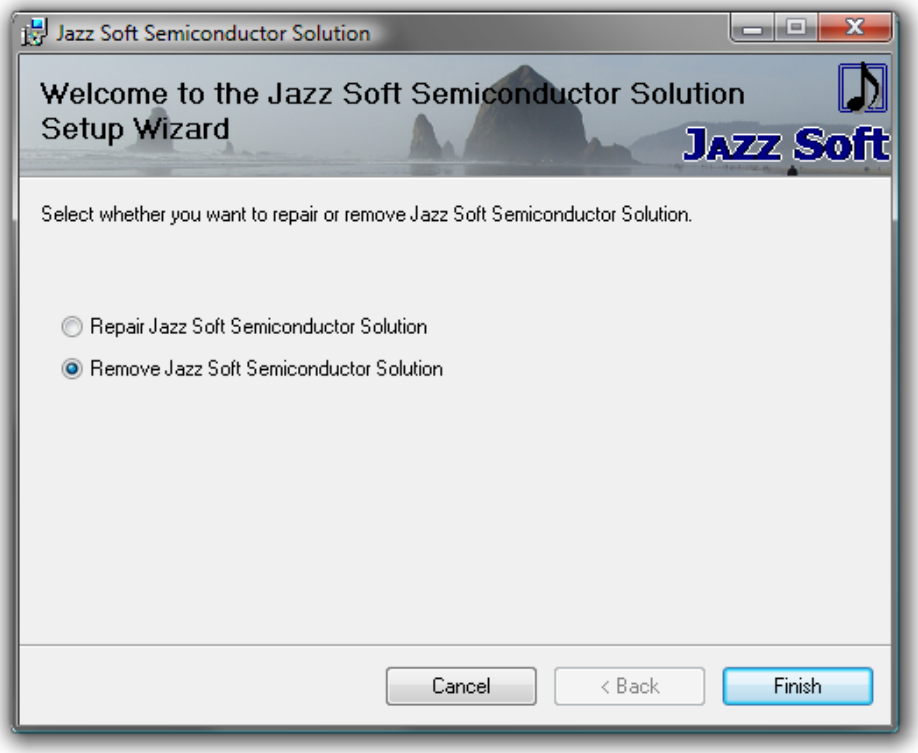
8 Once installation was completed successfully, following screen will appear. Click “Close” button.



4.3Uninstall

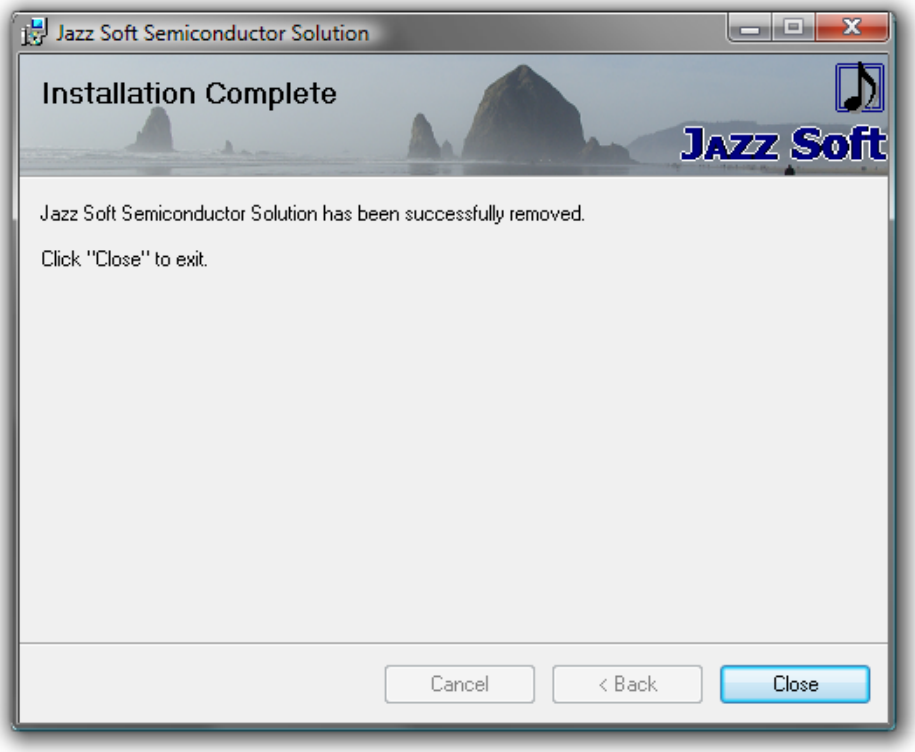
Uninstallation may be available in “Programs and Features” from Control Panel. Or rerun setup.exe again.

1 Run setup.exe (the one you installed). Choose “Remove Jazz Soft Semiconductor Solution” and click “Finish” button.



2 Actual uninstallation will start.

3 Once uninstallation was done successfully, following screen will appear. Click “Close” button.



5 Tutorial

5.1 Visual Basic 2008

This tutorial will explain programming with Savoy by making a mini host application.

5.1.1 Specification of Mini Host

Following is the specification of mini host which will control a general wafer inspection tool.

- Attempt on-line, select recipe, start inspection and collect data.
- The messages this software can send are as follows.

```
Select.req
Select.rsp
S1F13
S2F41
S6F12
```

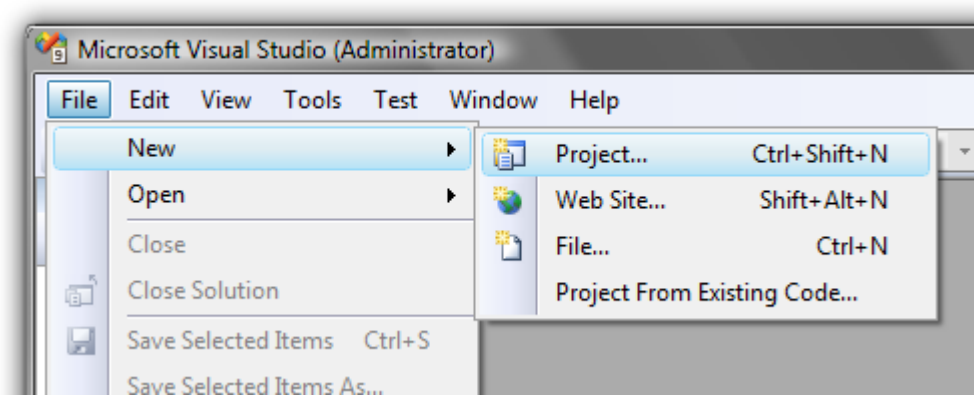
- The messages this software can receive are as follows.

```
Select.req
Select.rsp
S1F14
S2F42
S6F11
```

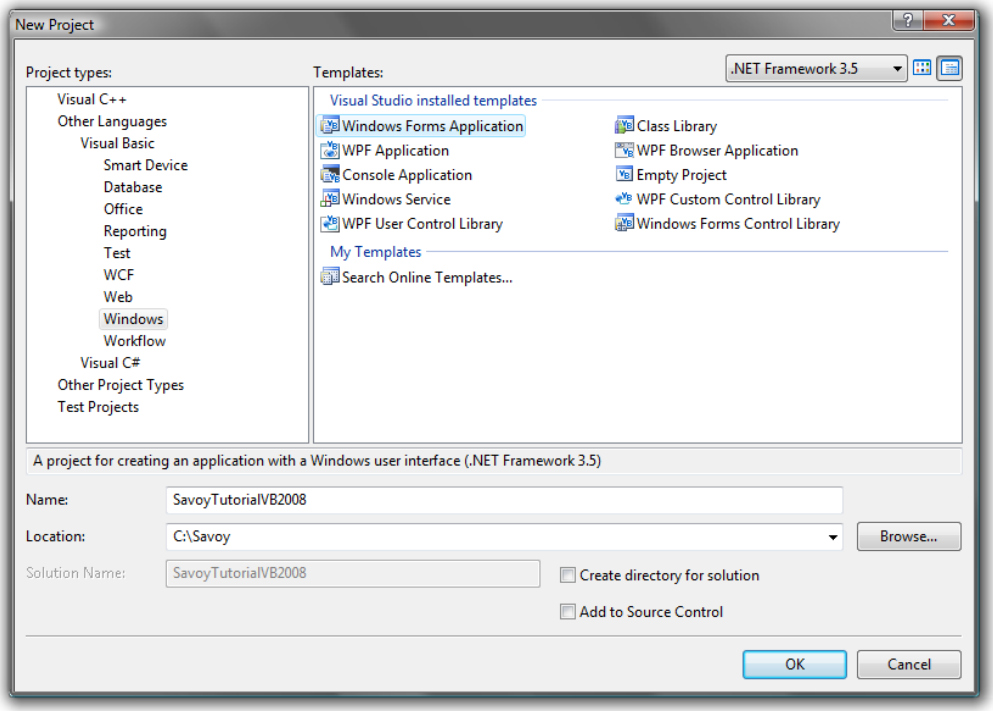
- Initial setting on equipment side had already been configured and done.
- Stream 9 and function 0 will not be processed.
- Don't check T3 timeout.

5.1.2 Create New Project

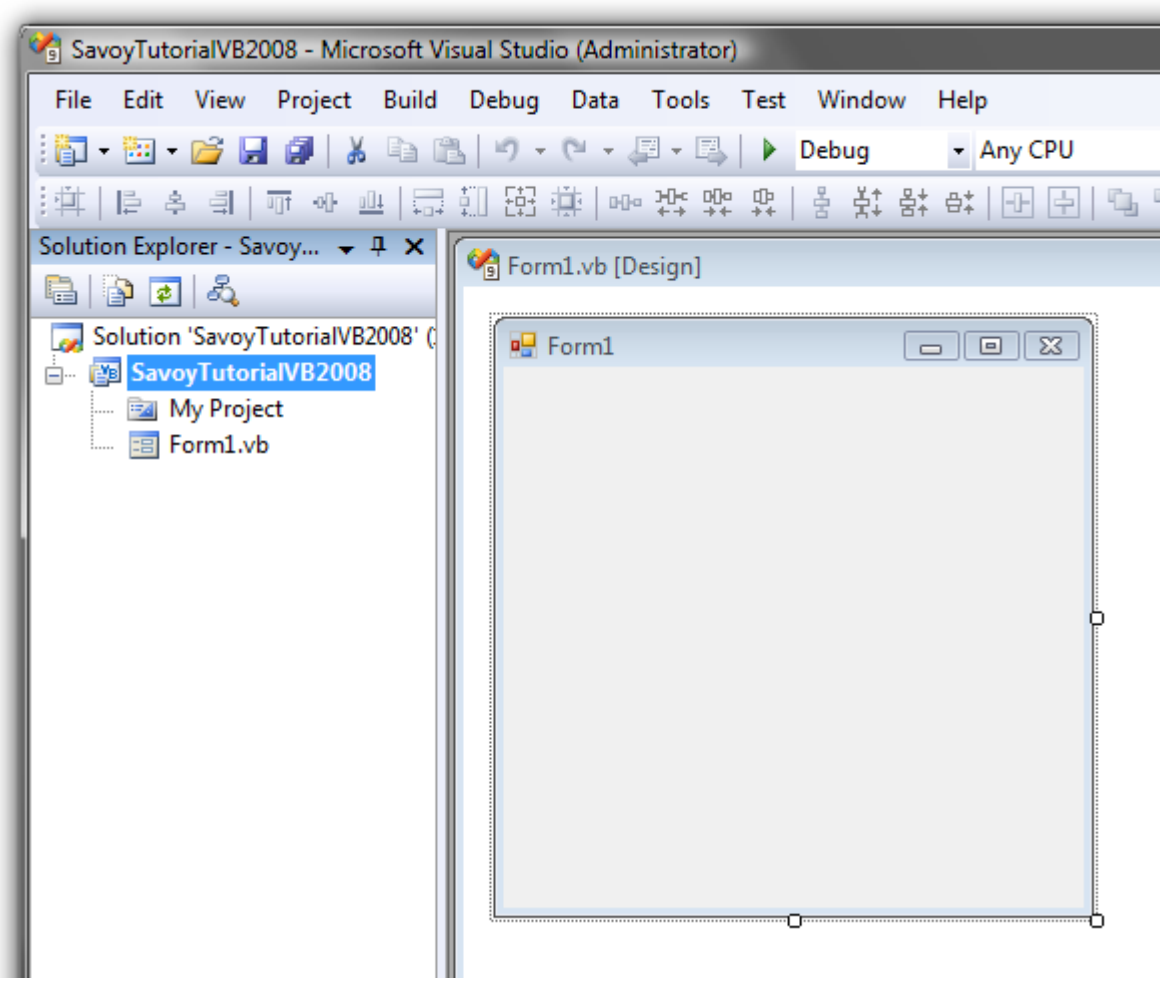
1 Launch Visual Studio 2008 and click [New] – [Project…] from File menu.



2 Choose "Windows Forms Application" from "Visual Basic" project type, and type project name and folder name. For example, project name could be "SavoyTutorialVB2008". If setting was OK, click "OK" button.



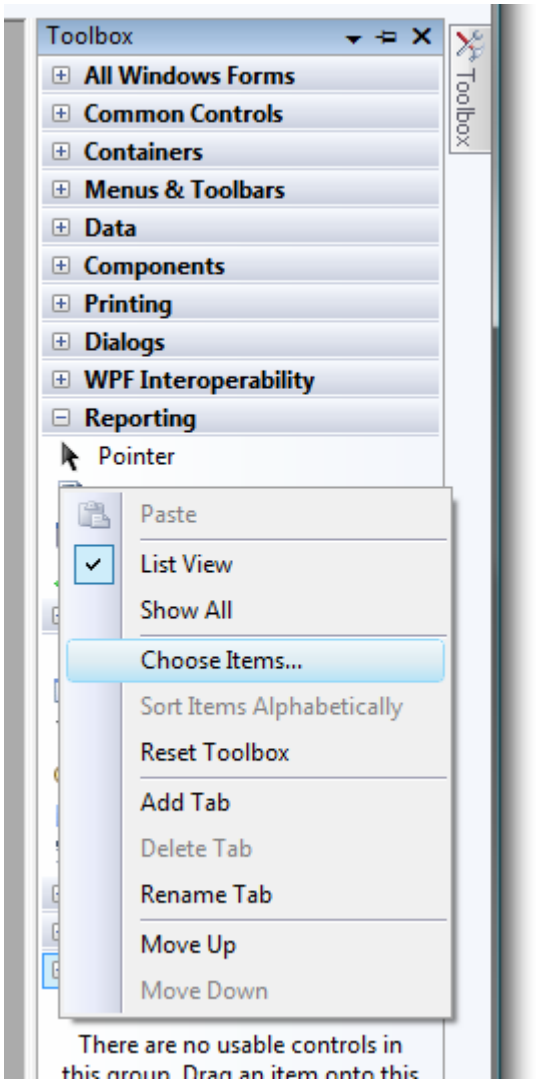
3 New project was created.



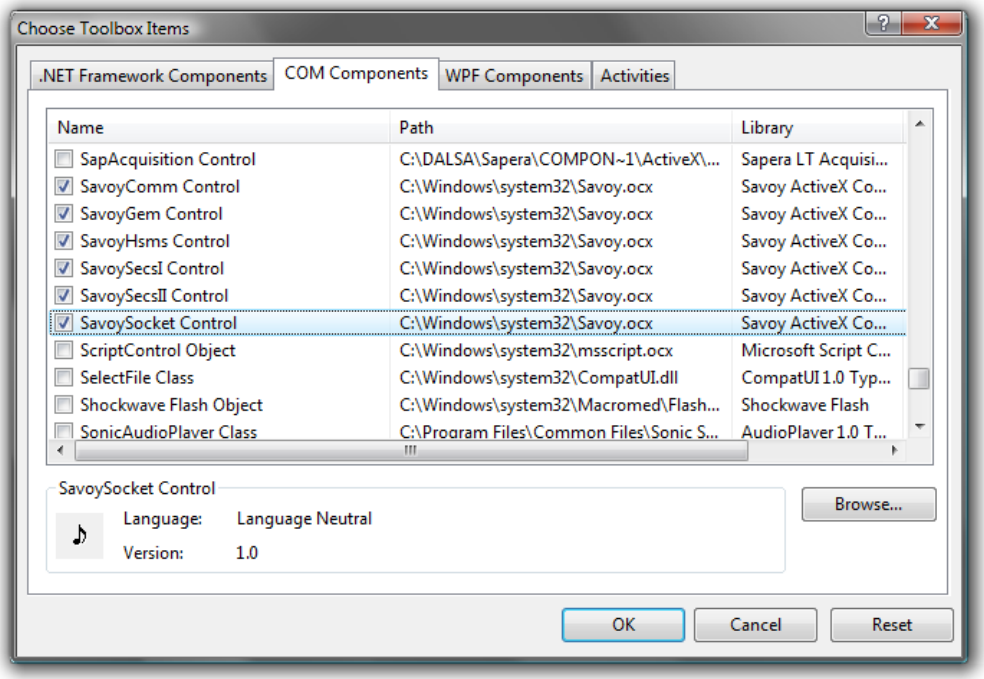
5.1.3 Add Savoy into Toolbox

Following procedure should be done only once. User doesn't have to do this again next time.

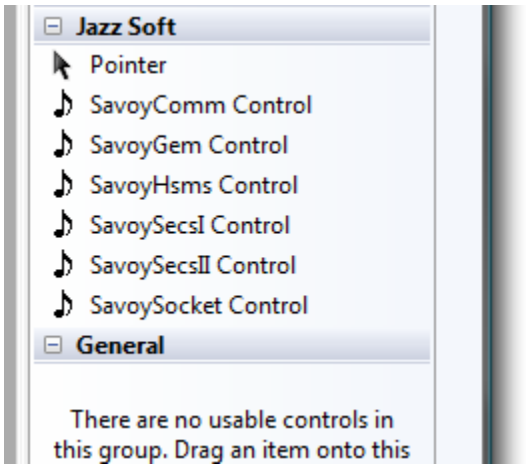
1 Right click on empty area on Toolbox, click "Choose Items…" from the popup menu. This may take more than one minute until the next screen will appear.



2 Select "COM Components" tab and put check mark on "Savoy ActiveX Control module". Click "OK" button.

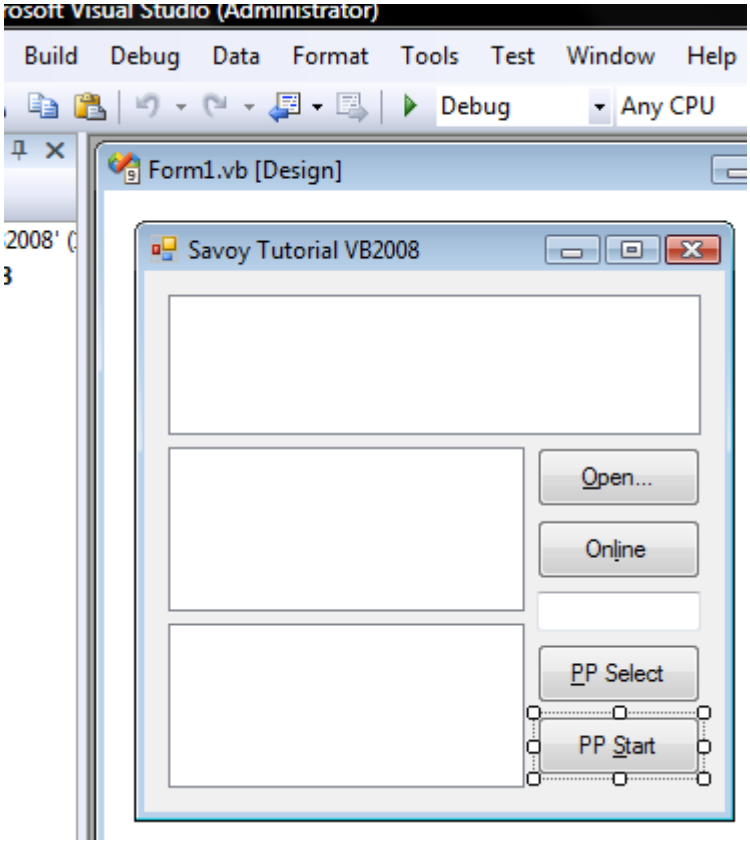


3 Since Savoy will be added to tool box, add new tab such as “Jazz Soft” and put them in it for ease of access in a future, if needed.



5.1.4 Paste Savoy into Form

1 Place 1 SavoyHsms and 2 SavoySecsIIs on the form as follows.



2 Make button event handler function. If user clicks “Open” button, communication setting dialog box of SavoyHsms will appear on the screen. And then establish connection, if user clicks “Open” button. Since communication setting would be saved in “Savoy.ini” file, user doesn’t have to change the setting next time.

Visual Basic 2008

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
    ' Setup
    hsms.LoadIniFile()
    If hsms.Setup("") Then
        ' If OK button was pressed, establish connection
        hsms.Connect = True
    End If
End Sub
```

3 If user clicks “Online” button, send S1F13 message.

Visual Basic 2008

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
    ' Send S1F13
    outmsg.SML = "s1f13w{}"
    hsms.Send(outmsg.Msg)
End Sub
```

4 If user clicks “PP Select” button, send remote command “PP-SELECT”.

Visual Basic 2008

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
    ' Send S2F41 PP-Select
    outmsg.SML = "s2f41w{<a'PP-SELECT'>{{<a'PPID'><a'" + TextBox1.Text + "'>}}}"
    hsms.Send(outmsg.Msg)
End Sub
```

5 If user clicks “PP Start” button, send remote command “START”.

Visual Basic 2008

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button4.Click
    ' Send S2F41 Start
    outmsg.SML = "s2f41w{<A'START'>{{}}}"
    hsms.Send(outmsg.Msg)
End Sub
```

5.1.5 Event Procedure

Capture events from SavoyHsms control.

1 If Connected event occurs, send select request message.

Visual Basic 2008

```
Private Sub hsms_Connected(ByVal sender As System.Object, ByVal e As
AxSAVOYLib._DSavoyHsmsEvents_ConnectedEvent) Handles hsms.Connected
    ' Connected
    ' Send select request
    outmsg.SML = "Select.req"
    hsms.Send(outmsg.Msg)
End Sub
```

2 If Received event occurs, pass incoming message to SavoySecsII control to analyze message structure.

Visual Basic 2008

```
Private Sub hsms_Received(ByVal sender As System.Object, ByVal e As
AxSAVOYLib._DSavoyHsmsEvents_ReceivedEvent) Handles hsms.Received
    inmsg.Msg = e.lpszMsg
```

3 If incoming message requires reply message, return appropriate message such as “<b 0>”.

Visual Basic 2008

```
    Select Case inmsg.SType
        Case 0
            ' Data message
```

```
        If inmsg.Wbit And (inmsg.Function Mod 2) <> 0 Then
            ' Need to reply something...
            outmsg.SML = "<b 0>"
            outmsg.Reply(e.lpszMsg)
            hsms.Send(outmsg.Msg)
        End If
```

4 If select request message arrives, reply select response message.

Visual Basic 2008

```
        Case 1
            ' Select request
            outmsg.SML = "Select.rsp"
            outmsg.Reply(e.lpszMsg)
            hsms.Send(outmsg.Msg)
```

5.1.6 Entire Source Code

That's it. This project was created from zero, however, the number of lines of entire source code is only 55 lines including empty lines and comments. Actual code we wrote was only 25 lines except comment lines. We didn't write any configuration file or data file which typically was required by competitors' products. Competitors' products are never as simple as Savoy.

Visual Basic 2008

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        ' Setup
        hsms.LoadIniFile()
        If hsms.Setup("") Then
            ' If OK button was pressed, establish connection
            hsms.Connect = True
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
        ' Send S1F13
        outmsg.SML = "s1f13w{}"
        hsms.Send(outmsg.Msg)
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
        ' Send S2F41 PP-Select
        outmsg.SML = "s2f41w{<a'PP-SELECT'>{{<a'PPID'><a'" + TextBox1.Text + "'>}}}"
        hsms.Send(outmsg.Msg)
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button4.Click
        ' Send S2F41 Start
        outmsg.SML = "s2f41w{<A'START'>{{}}}"
        hsms.Send(outmsg.Msg)
    End Sub

    Private Sub hsms_Connected(ByVal sender As System.Object, ByVal e As
AxSAVOYLib._DSavoyHsmsEvents_ConnectedEvent) Handles hsms.Connected
        ' Connected
        ' Send select request
```

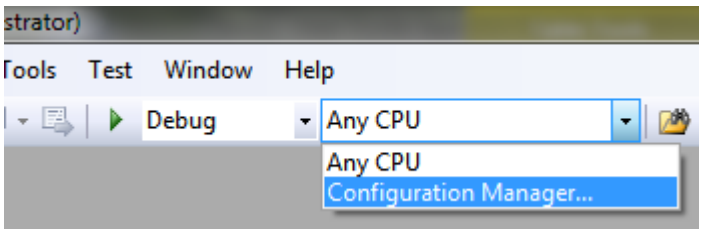
```
        outmsg.SML = "Select.req"
        hsms.Send(outmsg.Msg)
    End Sub

    Private Sub hsms_Received(ByVal sender As System.Object, ByVal e As
AxSAVOYLib._DSavoyHsmsEvents_ReceivedEvent) Handles hsms.Received
        inmsg.Msg = e.lpszMsg
        Select Case inmsg.SType
            Case 0
                ' Data message
                If inmsg.Wbit And (inmsg.Function Mod 2) <> 0 Then
                    ' Need to reply something...
                    outmsg.SML = "<b 0>"
                    outmsg.Reply(e.lpszMsg)
                    hsms.Send(outmsg.Msg)
                End If
            Case 1
                ' Select request
                outmsg.SML = "Select.rsp"
                outmsg.Reply(e.lpszMsg)
                hsms.Send(outmsg.Msg)
        End Select
    End Sub
End Class
```

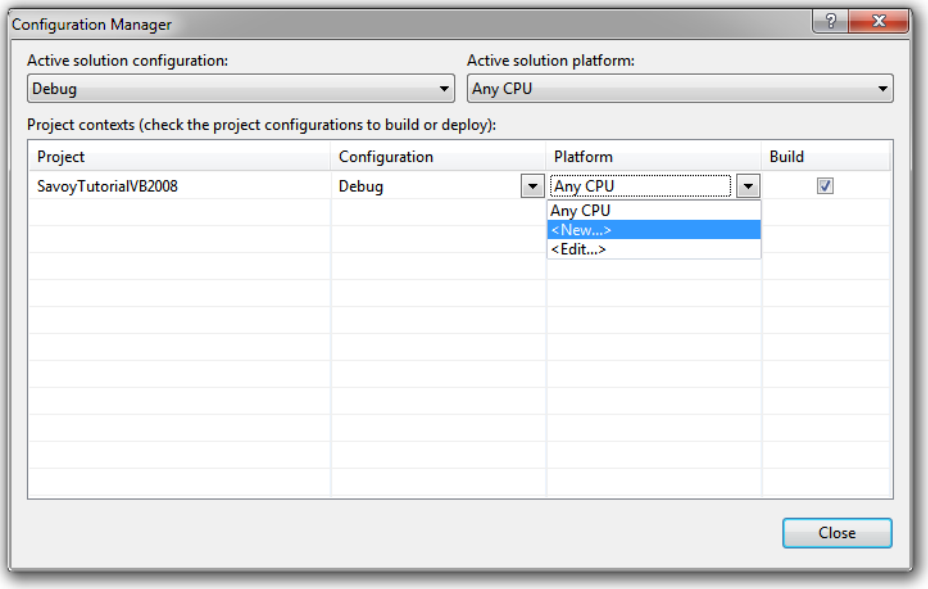
5.1.7 Important information for Windows 64-bit edition

It is required to set as x86 mode in order to run Windows 64-bit edition environment. Change project configuration as follows.

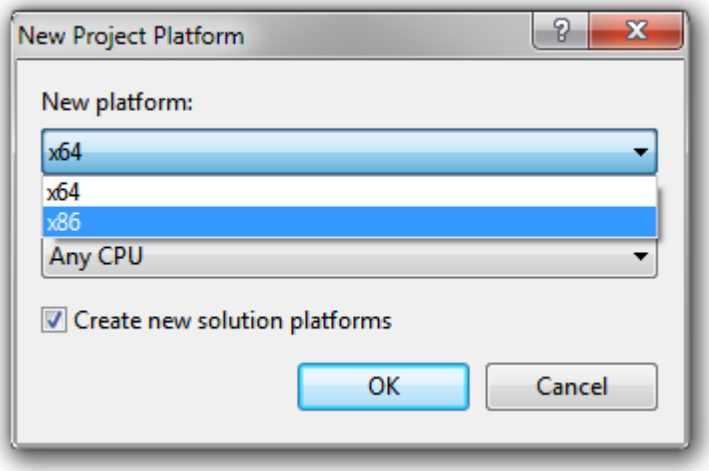
5 Click on Configuration Manager from tool bar.



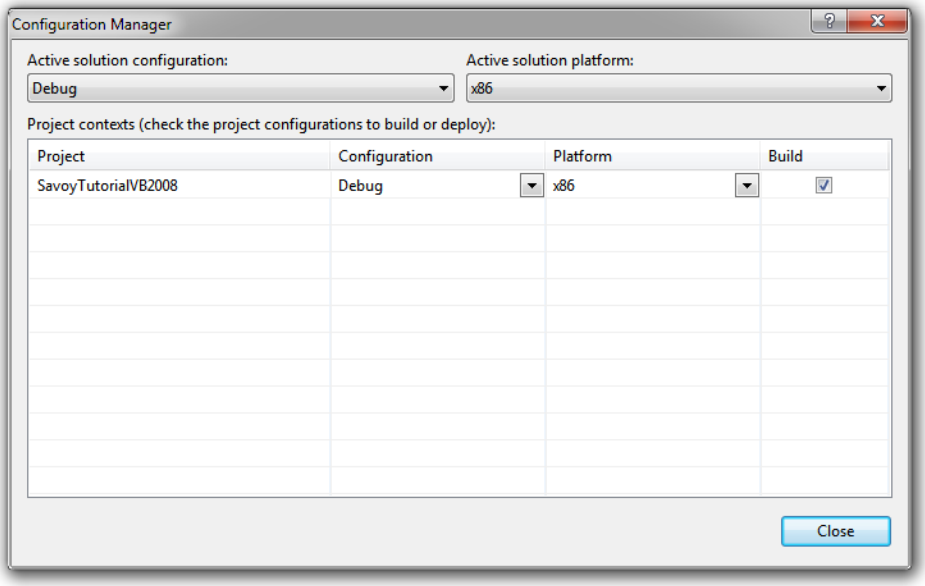
6 Select <New...> from Platform.



7 Select x86 from New platform and click OK button.



8 Check to make sure Platform is x86.



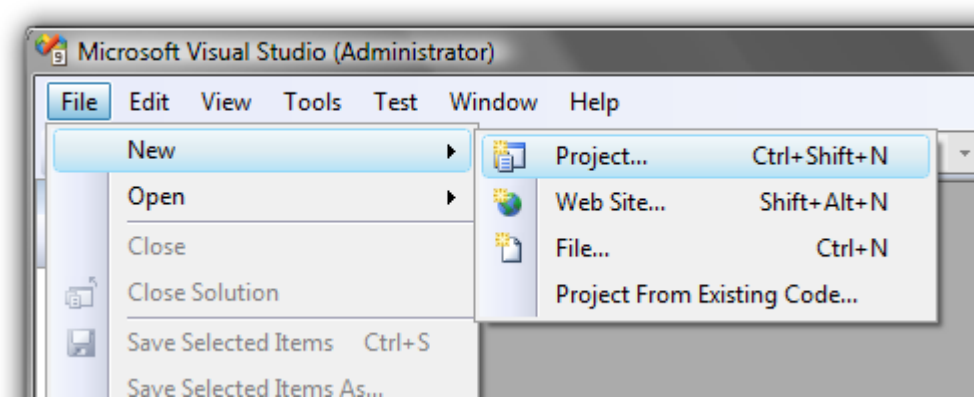
9 Above procedure is for Debug configuration. Do same thing for Release configuration.

5.2 C# 2008

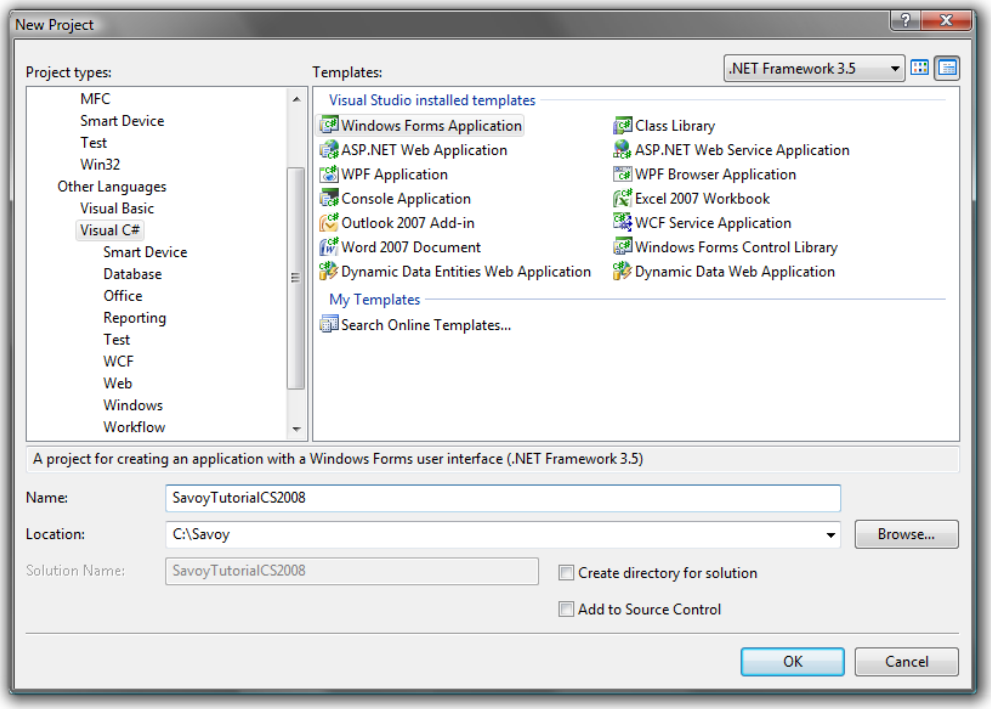
Application we created in previous chapter was written in Visual Basic 2008. Let's make the same application in C# 2008.

5.2.1 Create New Project

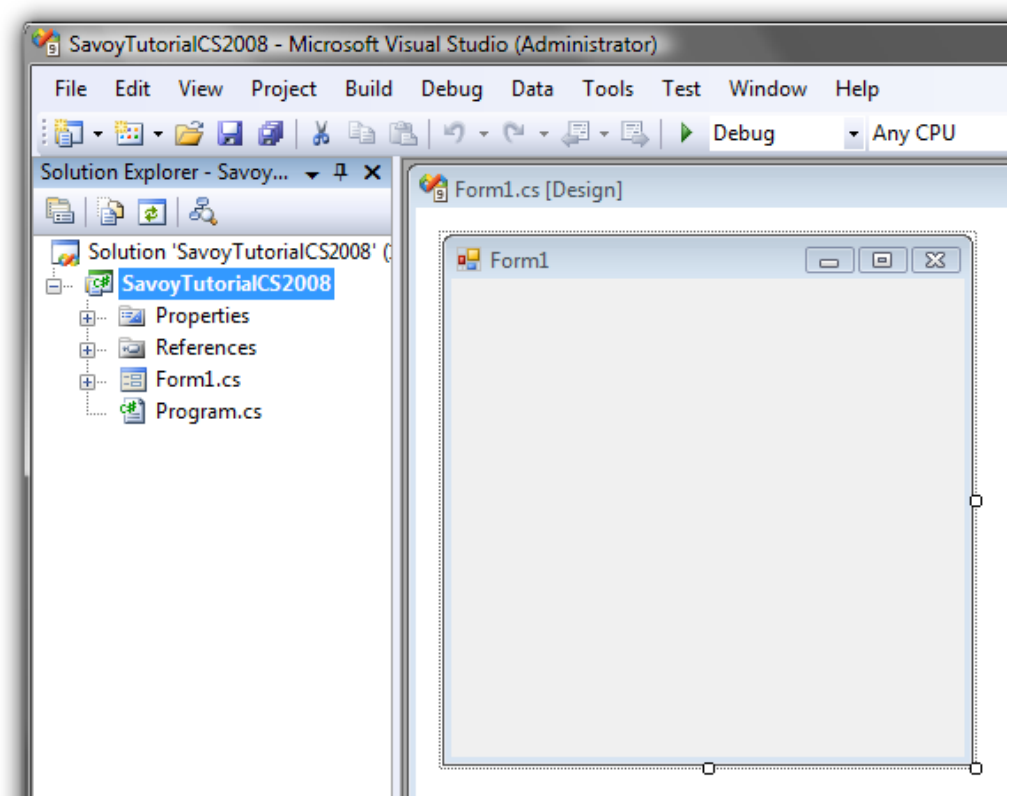
1 Launch Visual Studio 2008 and click [New] – [Project…] from File menu.



2 Choose "Windows Forms Application" from "Visual C#" project type, and type project name and folder name. For example, project name could be "SavoyTutorialCS2008". If setting was OK, click "OK" button.

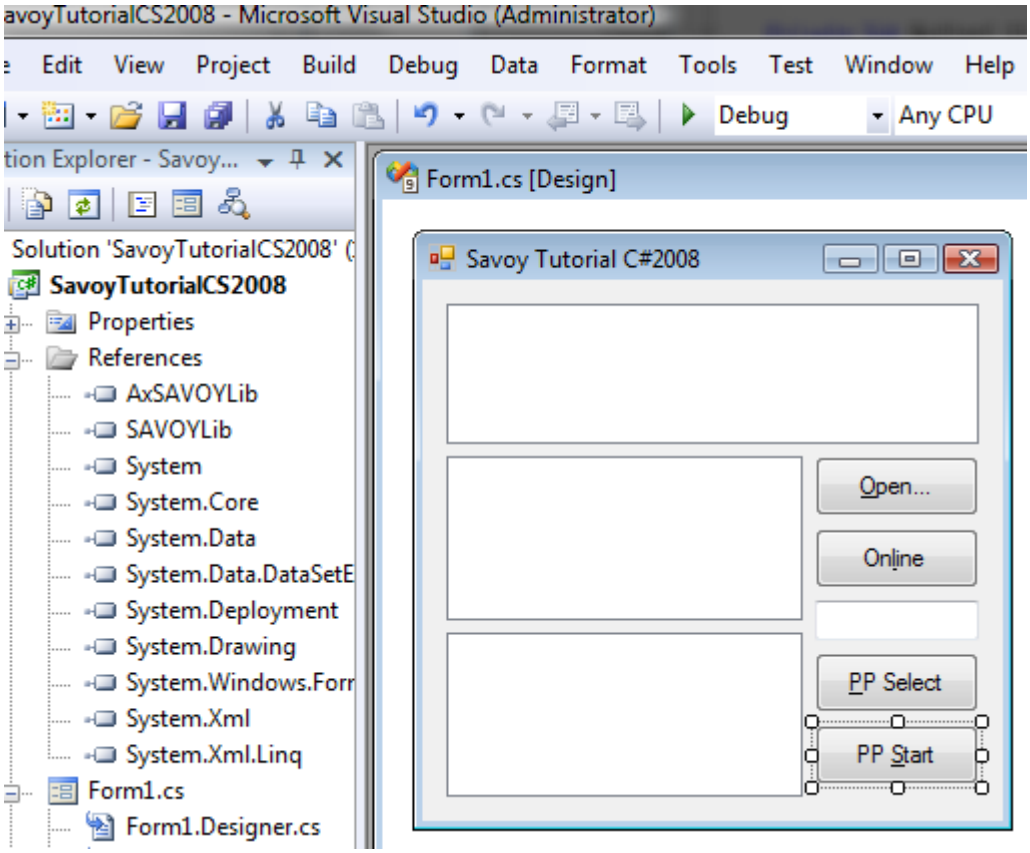


3 New project was created.



5.2.2 Paste Savoy into Form

1 Place 1 SavoyHsms and 2 SavoySecsIIs on the form as follows.



2 Make button event handler function. If user clicks “Open” button, communication setting dialog box of SavoyHsms will appear on the screen. And then establish connection, if user clicks “Open” button. Since communication setting would be saved in “Savoy.ini” file, user doesn’t have to change the setting next time.

Visual Basic 2008

```
private void button1_Click(object sender, EventArgs e)
{
    // Setup
    hsms.LoadIniFile();
    if (hsms.Setup(""))
    {
        // If OK button was pressed, establish connection
        hsms.Connect = true;
    }
}
```

3 If user clicks “Online” button, send S1F13 message.

Visual Basic 2008

```
private void button2_Click(object sender, EventArgs e)
{
    // Send S1F13
    outmsg.SML = "s1f13w{}";
    hsms.Send(outmsg.Msg);
}
```

4 If user clicks “PP Select” button, send remote command “PP-SELECT”.

Visual Basic 2008

```
private void button3_Click(object sender, EventArgs e)
{
    // Send S2F41 PP-Select
    outmsg.SML = "s2f41w{<a'PP-SELECT'>{{<a'PPID'><a'" + textBox1.Text + "'>}}}";
    hsms.Send(outmsg.Msg);
}
```

5 If user clicks “PP Start” button, send remote command “START”.

Visual Basic 2008

```
private void button4_Click(object sender, EventArgs e)
{
    // Send S2F41 Start
    outmsg.SML = "s2f41w{<A'START'>{{}}}";
    hsms.Send(outmsg.Msg);
}
```

5.2.3 Event Procedure

Capture events from SavoyHsms control.

1 If Connected event occurs, send select request message.

Visual Basic 2008

```
private void hsms_Connected(object sender, AxSAVOYLib._DSavoyHsmsEvents_ConnectedEvent e)
{
    // Connected
    // Send select request
    outmsg.SML = "Select.req";
    hsms.Send(outmsg.Msg);
}
```

2 If Received event occurs, pass incoming message to SavoySecsII control to analyze message structure.

Visual Basic 2008

```
private void hsms_Received(object sender, AxSAVOYLib._DSavoyHsmsEvents_ReceivedEvent e)
{
    inmsg.Msg = e.lpszMsg;
```

3 If incoming message requires reply message, return appropriate message such as “<b 0>”.

Visual Basic 2008

```
switch(inmsg.SType)
{
case 0:
    // Data message
    if(inmsg.Wbit && (inmsg.Function % 2)!=0)
    {
        // Need to reply something...
        outmsg.SML = "<b 0>";
        outmsg.Reply(e.lpszMsg);
        hsms.Send(outmsg.Msg);
    }
    break;
```

4 If select request message arrives, reply select response message.

Visual Basic 2008

```
case 1:
    // Select request
    outmsg.SML = "Select.rsp";
    outmsg.Reply(e.lpszMsg);
    hsms.Send(outmsg.Msg);
    break;
```

5.2.4 Entire Source Code

That’s it. This project was created from zero, however, the number of lines of entire source code is only 83 lines including empty lines and comments. Actual code we wrote was only 30 lines except comment lines. We

didn't write any configuration file or data file which typically was required by competitors' products. Competitors' products are never as simple as Savoy.

Visual Basic 2008

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

namespace SavoyTutorialCS2008
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Setup
            hsms.LoadIniFile();
            if (hsms.Setup(""))
            {
                // If OK button was pressed, establish connection
                hsms.Connect = true;
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            // Send S1F13
            outmsg.SML = "s1f13w{}";
            hsms.Send(outmsg.Msg);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            // Send S2F41 PP-Select
            outmsg.SML = "s2f41w{<a'PP-SELECT'>{{<a'PPID'><a'" + textBox1.Text + "'>}}}";
            hsms.Send(outmsg.Msg);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            // Send S2F41 Start
            outmsg.SML = "s2f41w{<A'START'>{{}}}";
            hsms.Send(outmsg.Msg);
        }

        private void hsms_Connected(object sender, AxSAVOYLib._DSavoyHsmsEvents_ConnectedEvent e)
        {
            // Connected
            // Send select request
            outmsg.SML = "Select.req";
            hsms.Send(outmsg.Msg);
        }

        private void hsms_Received(object sender, AxSAVOYLib._DSavoyHsmsEvents_ReceivedEvent e)
        {
            inmsg.Msg = e.lpszMsg;
            switch(inmsg.SType)
            {
```

```
                case 0:
                    // Data message
                    if(inmsg.Wbit && (inmsg.Function % 2)!=0)
                    {
                        // Need to reply something...
                        outmsg.SML = "<b 0>";
                        outmsg.Reply(e.lpszMsg);
                        hsms.Send(outmsg.Msg);
                    }
                    break;
                case 1:
                    // Select request
                    outmsg.SML = "Select.rsp";
                    outmsg.Reply(e.lpszMsg);
                    hsms.Send(outmsg.Msg);
                    break;
                }
            }
        }
    }
```

5.2.5 Important information for Windows 64-bit edition

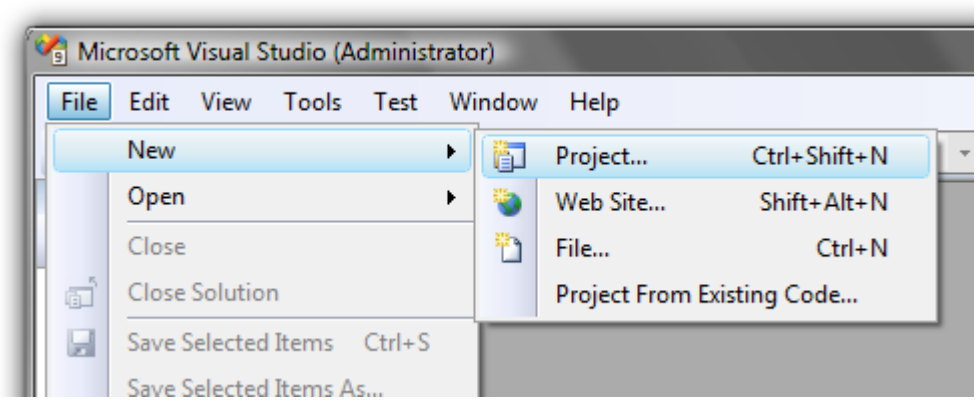
It is required to set as x86 mode in order to run Windows 64-bit edition environment. Refer to above Visual Basic 2008 sample and change project configuration.

5.3 Visual C++ 2008

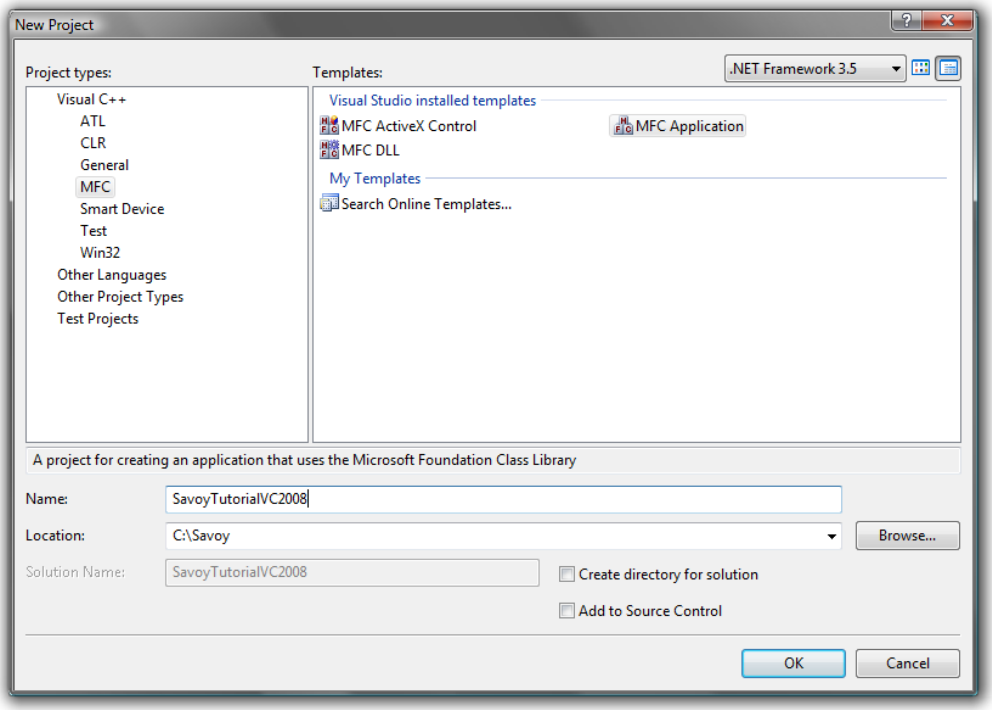
Development procedure and even source code of software for Visual Basic 2008 and C# 2008 was quite similar. But there is much difference for VC++ 2008.

5.3.1 Create New Project

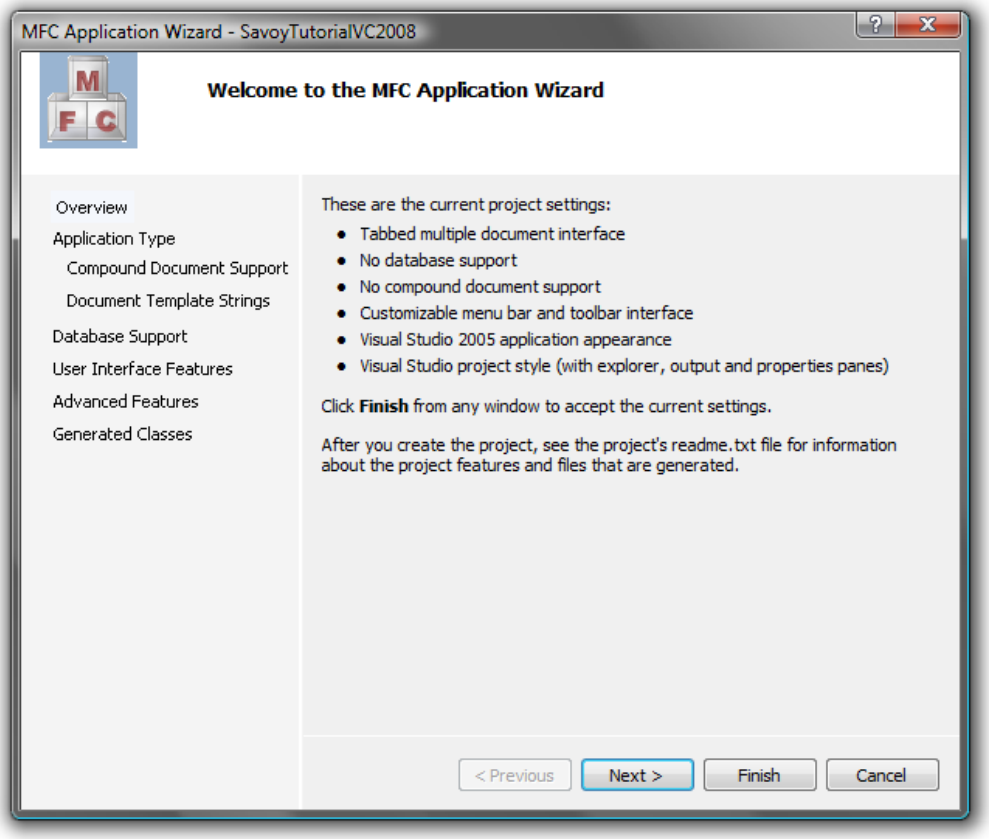
1 Launch Visual Studio 2008 and click [New] – [Project…] from File menu.



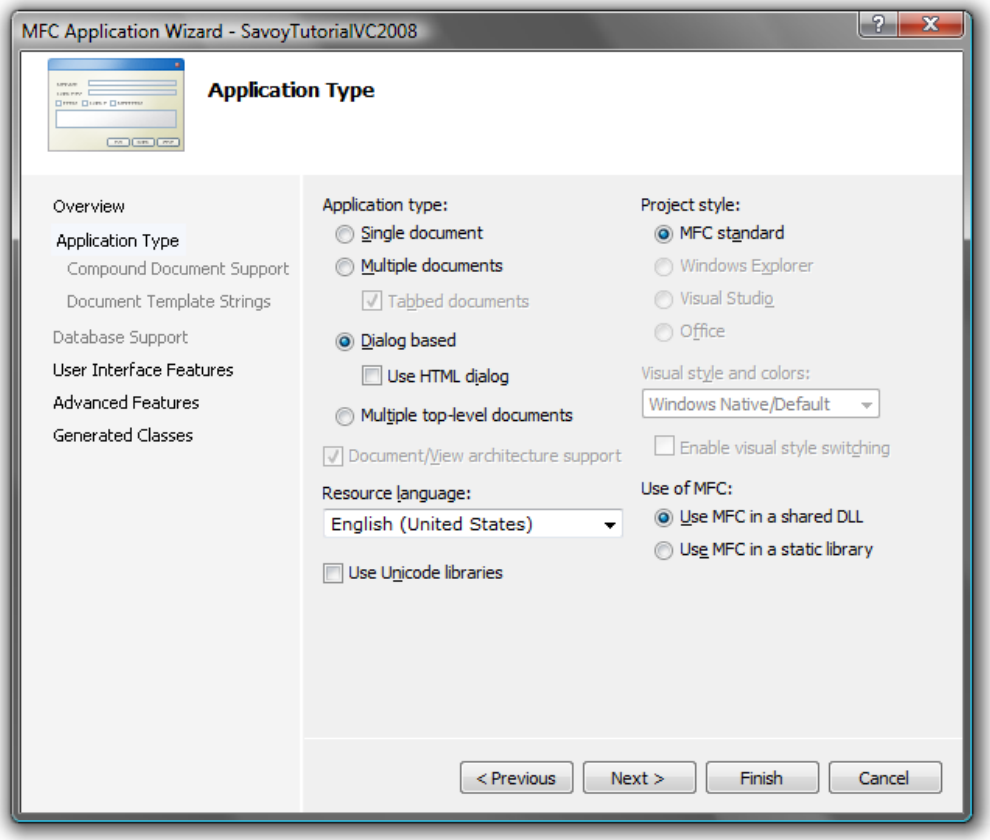
2 Choose “MFC Application” from “Visual C++” project type, and type project name and folder name. For example, project name could be “SavoyTutorialVC2008”. If setting was OK, click “OK” button.



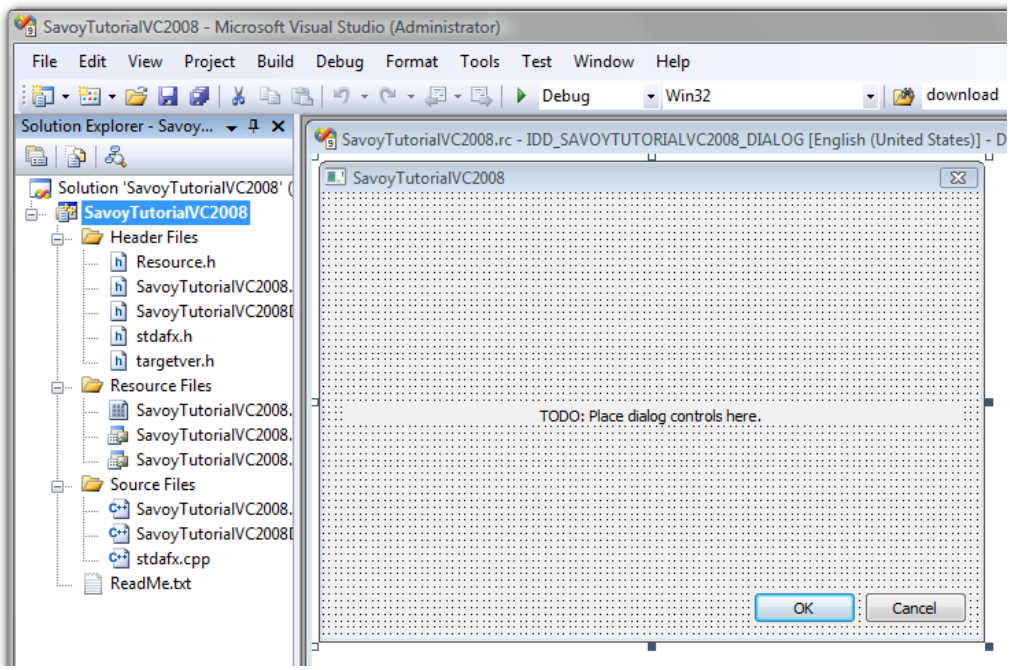
3 Welcome screen will appear. Click “Next” button.



4 Choose “Dialog based” application and uncheck “Use Unicode libraries”. Since other settings are OK by default, click “Finish” button.

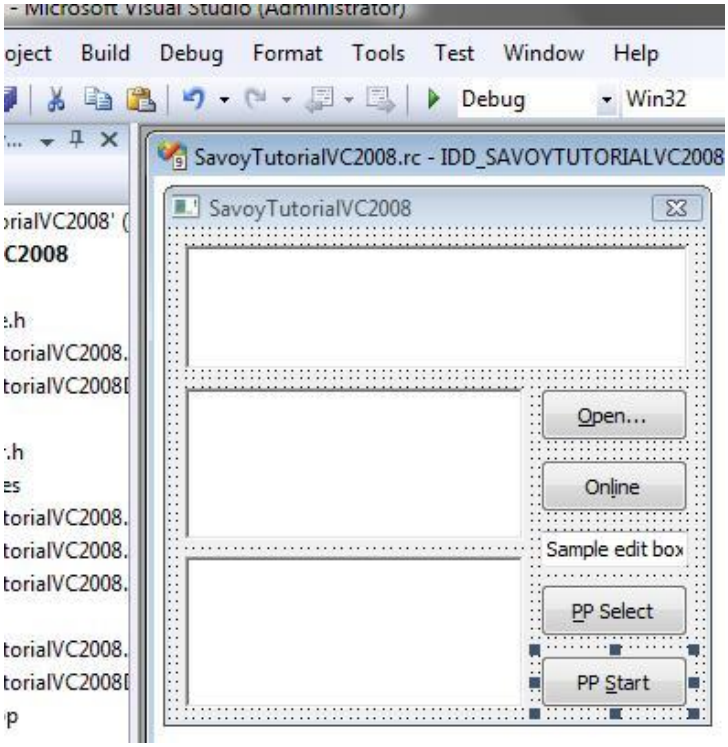


5 New project was created.

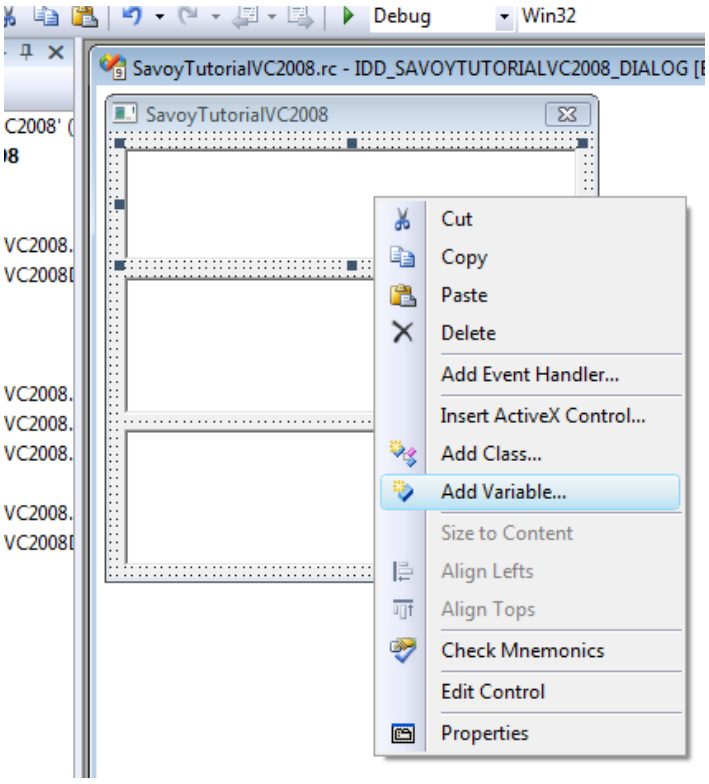


5.3.2 Paste Savoy into Form

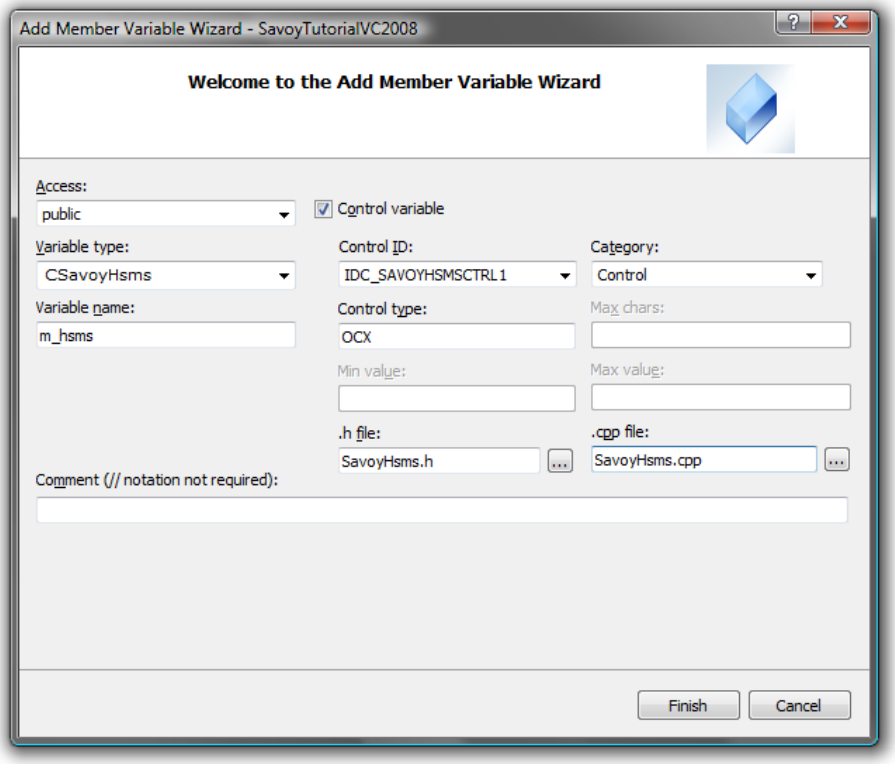
1 Place 1 SavoyHsms and 2 SavoySecsIIs on the dialog resource as follows.



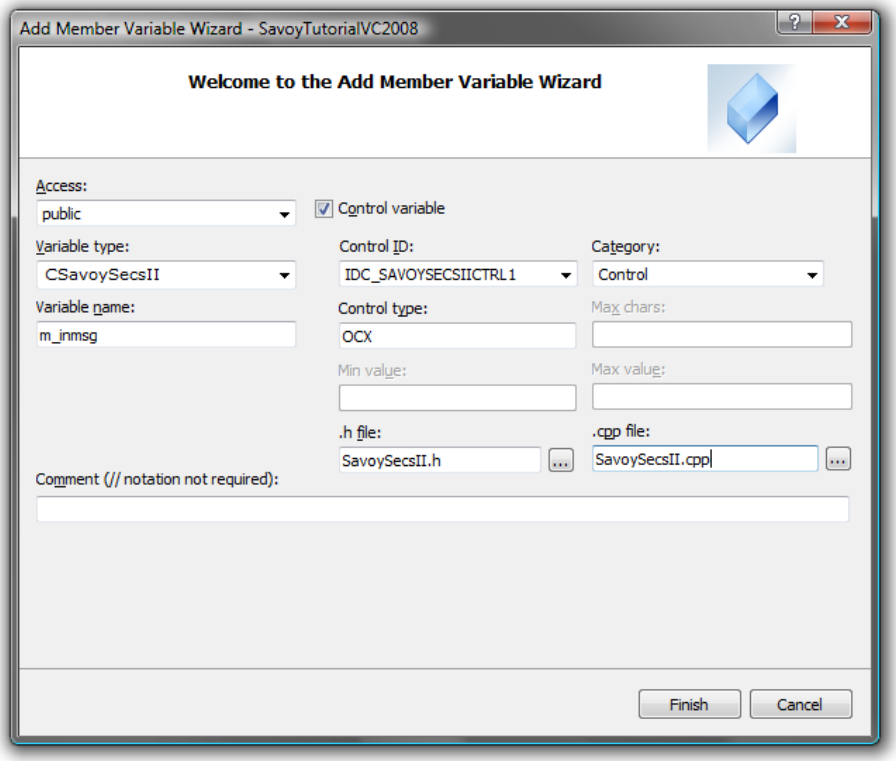
2 Assign as variable. Right click on SavoyHsms object, and click “Add Variable” from popup menu.



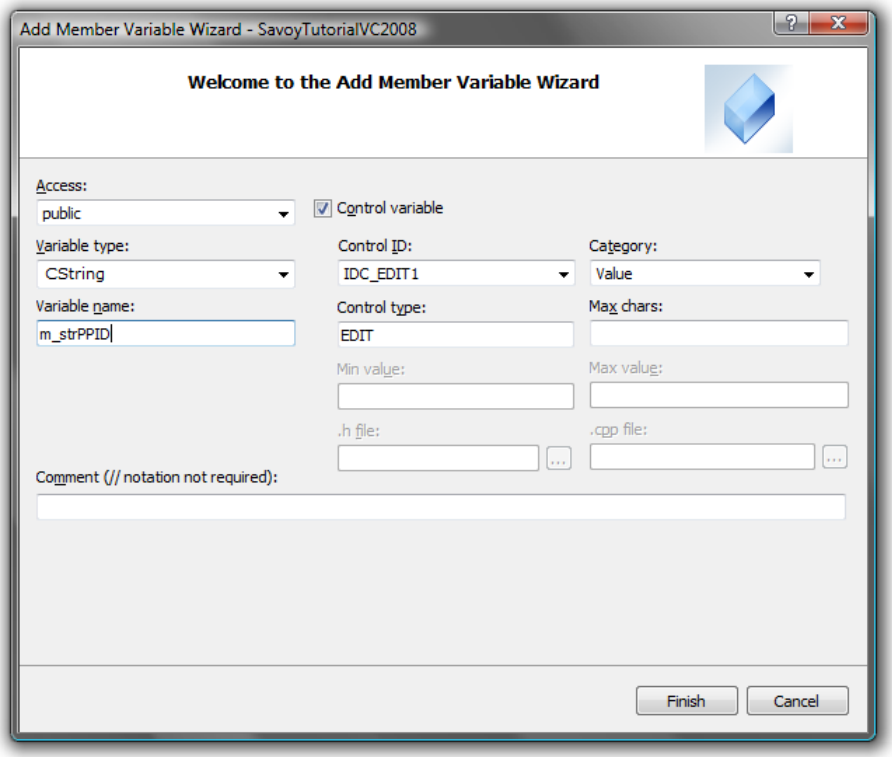
3 Change variable type to “CSavoyHsms”. The name of “.h file” and “.cpp file” will also be “SavoyHsms.h” and “SavoyHsms.cpp” respectively. Variable name will be “m_hsms”. Click “Finish” button.



4 Do same thing for SavoySecsII control. Change variable type to “CSavoySecsII”. The name of “.h file” and “.cpp file” will be “SavoySecsII.h” and “SavoySecsII.cpp” respectively. Variable name will be “m_inmsg” and “m_outmsg” respectively.



5 Variable for recipe name will be CString type and “m_strPPID”.



5.3.3 Overwrite Wrapper Classes

Visual C++ 2008 has some issues regarding ActiveX control wrapper class. So Jazz Soft provided wrapper class to solve this issue. Wrapper class could be made by Visual C++ 6.0, but “enum” is not reflected.

1 Close Visual C++ 2008 at this time.

2 Overwrite newly created “SavoyHsms.h”, “SavoyHsms.cpp”, “SavoySecsII.h” and “SavoySecsII.cpp” with ones provided by Jazz Soft.

3 Restart Visual C++ 2008 and reload project.

5.3.4 Process Buttons

1 Make button event handler function. If user clicks “Open” button, communication setting dialog box of SavoyHsms will appear on the screen. And then establish connection, if user clicks “Open” button. Since communication setting would be saved in “Savoy.ini” file, user doesn’t have to change the setting next time.

Visual Basic 2008

```
void CSavoyTutorialVC2008Dlg::OnBnClickedButton1()
{
    // Setup
    m_hsms.LoadIniFile();
    if(m_hsms.Setup(""))
    {
```

```
            // If OK button was pressed, establish connection
            m_hsms.SetConnect(true);
        }
}
```

2 If user clicks “Online” button, send S1F13 message.

Visual Basic 2008

```
void CSavoyTutorialVC2008Dlg::OnBnClickedButton2()
{
    // Send S1F13
    m_outmsg.SetSml("s1f13w{}");
    m_hsms.Send(m_outmsg.GetMsg());
}
```

3 If user clicks “PP Select” button, send remote command “PP-SELECT”.

Visual Basic 2008

```
void CSavoyTutorialVC2008Dlg::OnBnClickedButton3()
{
    // Send S2F41 PP-Select
    UpdateData();
    m_outmsg.SetSml("s2f41w{<a'PP-SELECT'>{{<a'PPID'><a'" + m_strPPID + "'>}}}");
    m_hsms.Send(m_outmsg.GetMsg());
}
```

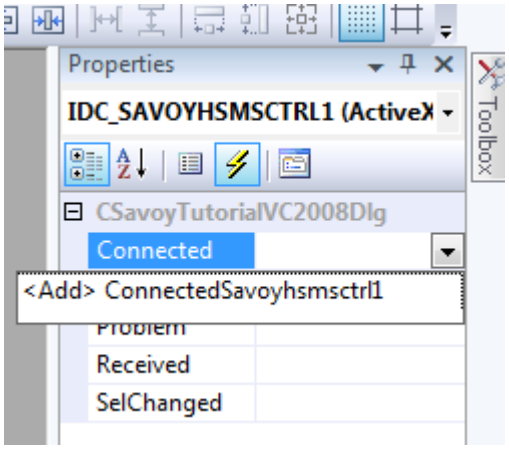
4 If user clicks “PP Start” button, send remote command “START”.

Visual Basic 2008

```
void CSavoyTutorialVC2008Dlg::OnBnClickedButton4()
{
    // Send S2F41 Start
    m_outmsg.SetSml("s2f41w{<A'START'>{{}}}");
    m_hsms.Send(m_outmsg.GetMsg());
}
```

5.3.5 Event Procedure

Capture events from SavoyHsms control. Event handler function can be created from “Properties” window.



1 If Connected event occurs, send select request message.

Visual Basic 2008

```
void CSavoyTutorialVC2008Dlg::ConnectedSavoyhsmsctrl1(LPCTSTR lpszIPAddress, long lPortNumber)
{
    // Connected
    // Send select request
    m_outmsg.SetSml("Select.req");
    m_hsms.Send(m_outmsg.GetMsg());
}
```

2 If Received event occurs, pass incoming message to SavoySecsII control to analyze message structure.

Visual Basic 2008

```
void CSavoyTutorialVC2008Dlg::ReceivedSavoyhsmsctrl1(LPCTSTR lpszIPAddress, long lPortNumber,
LPCTSTR lpszMsg)
{
    m_inmsg.SetMsg(lpszMsg);
```

3 If incoming message requires reply message, return appropriate message such as “<b 0>”.

Visual Basic 2008

```
    switch(m_inmsg.GetSType())
    {
    case 0:
        // Data message
        if(m_inmsg.GetWbit() && m_inmsg.GetFunction()%2)
        {
            // Need to reply something...
            m_outmsg.SetSml("<b 0>");
            m_outmsg.Reply(lpszMsg);
            m_hsms.Send(m_outmsg.GetMsg());
        }
        break;
```

4 If select request message arrives, reply select response message.

Visual Basic 2008

```
    case 1:
        // Select request
        m_outmsg.SetSml("Select.rsp");
        m_outmsg.Reply(lpszMsg);
        m_hsms.Send(m_outmsg.GetMsg());
        break;
    }
}
```

5.3.6 Entire Source Code

That's it. Since the entire source code consists of multiple files, I don't put them here.

This project was created from zero, however, the number of lines in core source code "SavoyTutorialVC2008Dlg.cpp" is only 231 lines including empty lines and comments. Actual code we wrote was only 31 lines except comment lines. We didn't write any configuration file or data file which typically was required by competitors' products. Competitors' products are never as simple as Savoy.