Savoy ActiveX コントロール
ユーザーガイド

Jazz Soft, Inc.

1　改訂履歴

バージョン	日付	氏名	説明
1.00	2009 年 7 月 31 日	Hikaru Okada	新規に作成。
1.00a	2009 年 8 月 22 日	Hikaru Okada	マニュアルのページ数が大きくなったため分割。
1.00c	2009 年 12 月 25 日	Hikaru Okada	Windows Vista および Windows 7 における注意点を追加。
1.00d	2010 年 12 月 18 日	Hikaru Okada	インストール時の製品選択画面を追加。 Windows 64 ビット版対応。
1.00e	2014 年 7 月 13 日	Carl Hikaru Okada	Windows 8.1 の記述を追加。

2 目次

3　使用環境

- Windows 2000, Windows XP, Windows Vista または Windows 7
- Visual Basic や Visual C++などの Active X 対応の開発言語

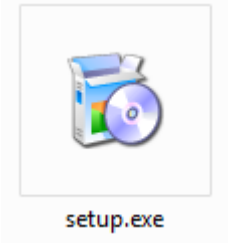
4　セットアップ

Windows には Administrator の権限でログインしてください。もし Windows Vista、Windows 7、Windows 8 もしくは Windows 8.1 をお使いでユーザアカウントコントロール（UAC）が有効になっている場合、インストールに失敗することがあります。一時的に UAC を無効にしてください。

開発環境も実行環境も同じです。Setup.exe を実行してインストールします。

1 旧バージョンの Jazz Soft Semiconductor Solution がインストールされている場合は、先にアンインストールしてください。アンインストールはコントロールパネルの「Programs and Features」から行えます。

2 Setup.exe を実行します。Microsoft Visual C++ 2008 SP1 用コンポーネント群がインストールされていない場合は、最初にそれらがインストールされます。同様に.NET Framework 3.5 がインストールされていない場合も、それらをインストールするよう表示される場合があります。



3 Jazz Soft Semiconductor Solution のセットアップ画面が表示されます。Next ボタンをクリックします。

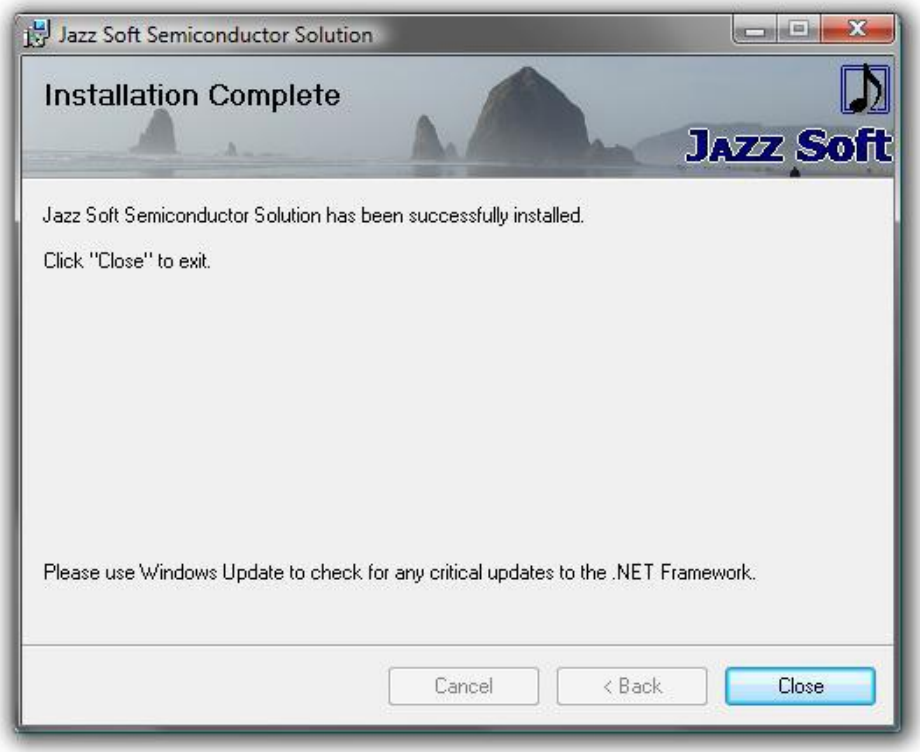
4 インストールするフォルダとユーザアカウントを選択します。通常は何も変更しないで Next ボタンをクリックします。

5 インストールする製品を選択します。Next ボタンをクリックします。

6 インストール前の確認画面です。Next ボタンをクリックします。

7 インストールが始まります。

8 インストールが正常に完了すると下記の画面になります。Close ボタンをクリックします。

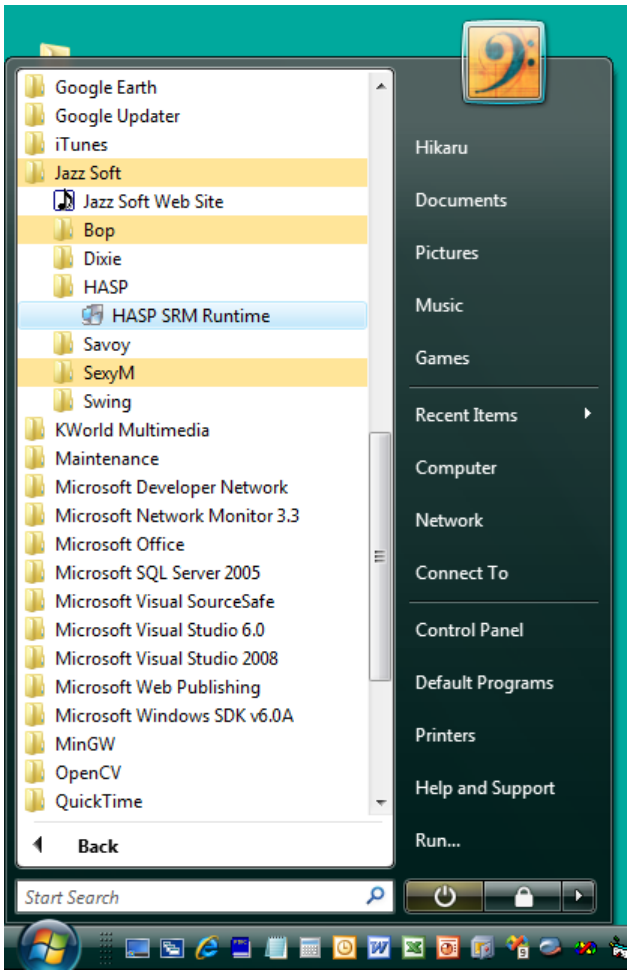


4.1HASP ドライバのインストール

HASP キー用のドライバは試用版では必要ありませんが、製品版として使用する場合には必要となります。HASP ドライバのセットアップに必要なソフトウェアは、Jazz Soft Semiconductor Solution のセットアップでコピーされます。

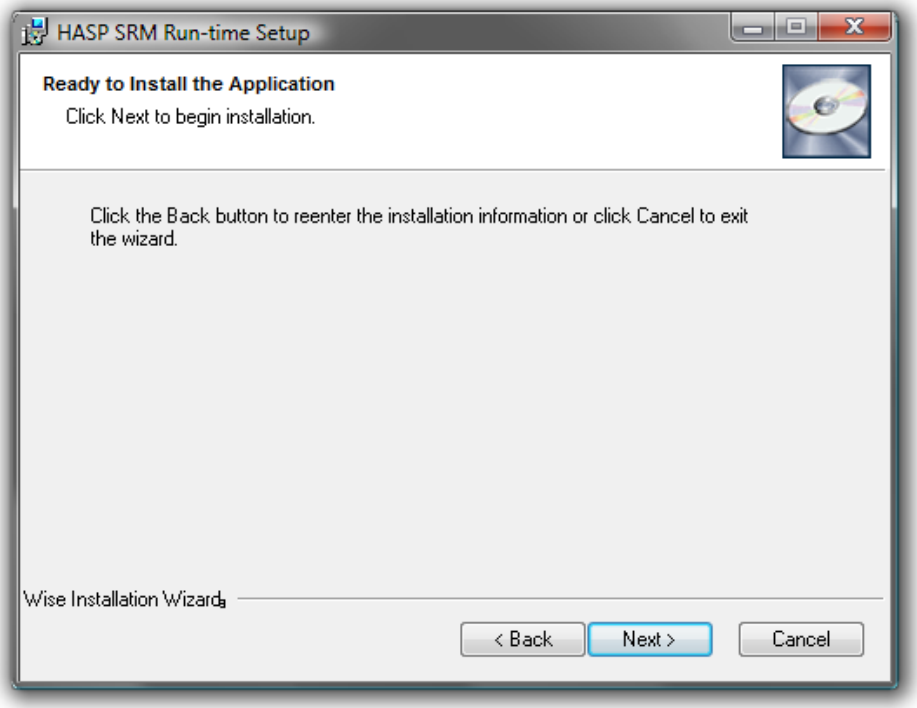
HASP ドライバをインストールする際には、HASP キーをコンピュータに挿さないでください。

9 スタートメニューの「Jazz Soft」 – 「HASP」から「HASP SRM Runtime」をクリックします。

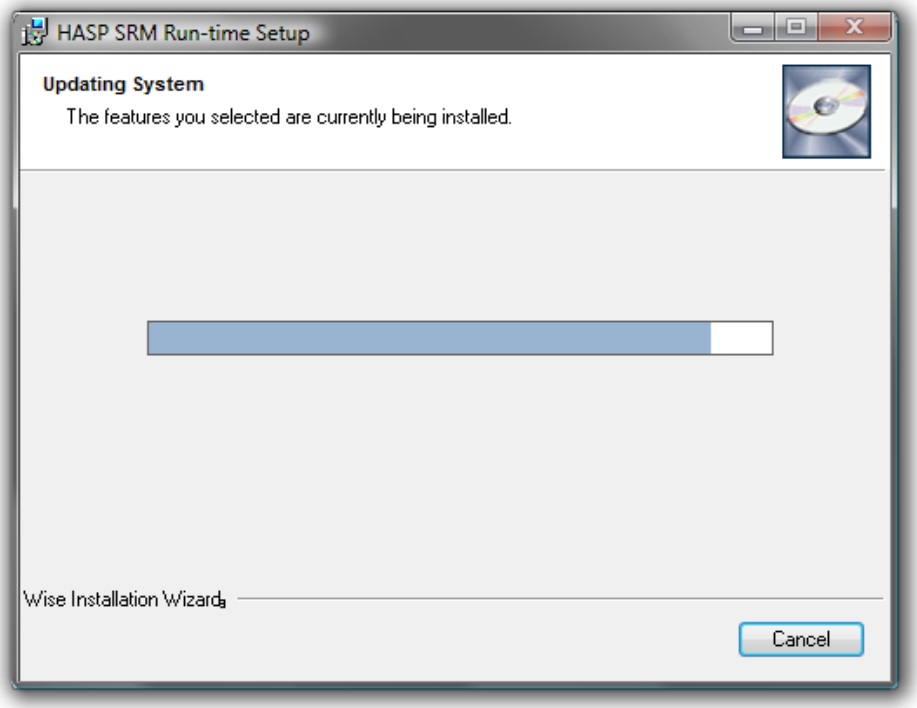


10 ウェルカム画面が表示されます。Next ボタンをクリックします。

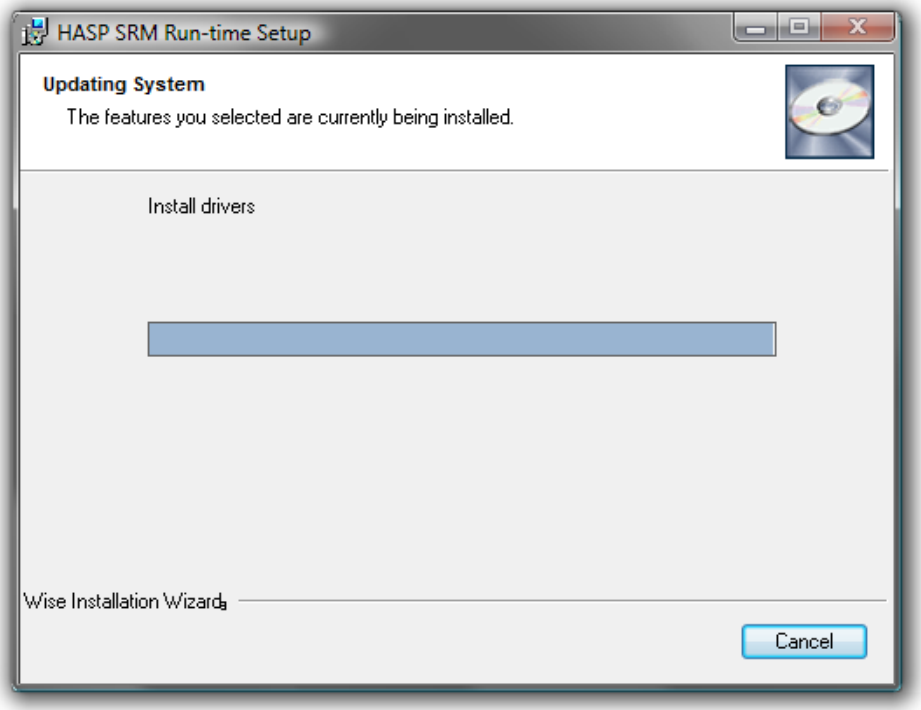
11 インストール前の確認が面が表示されます。Install ボタンをクリックします。



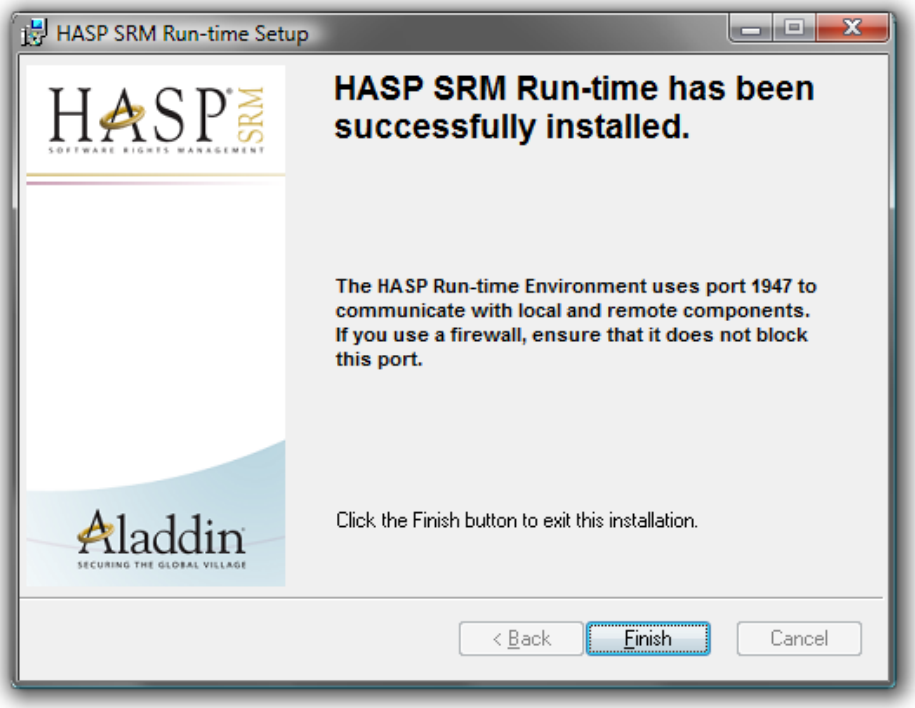
12インストールが始まります。



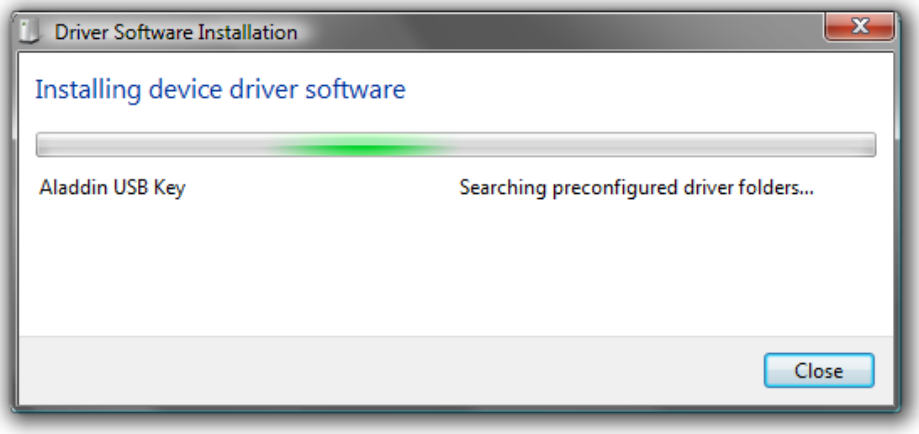
13「Install drivers」の表示のままで数分かかることがあります。



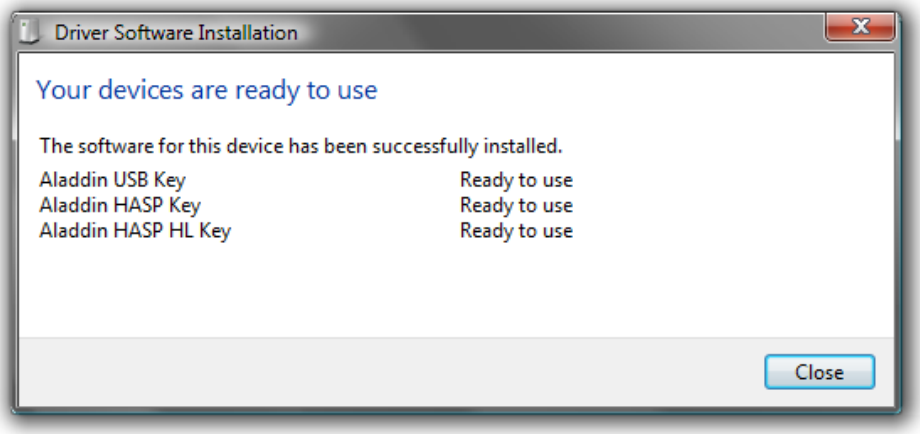
14ドライバのコピーが正常に完了すると下記の画面になります。Finish ボタンをクリックします。



15 ここで HASP キーをコンピュータに挿します。Windows Vista の場合は画面右下にドライバのインストール状況が表示されます。



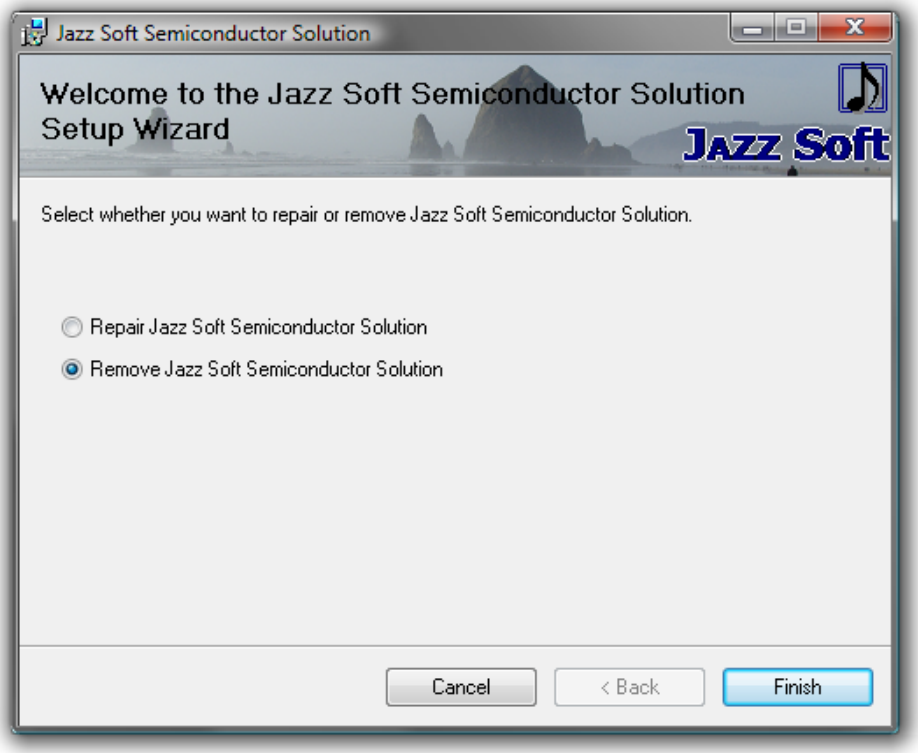
16 インストールが正常に完了すると下記の画面になります。Close ボタンをクリックします。



4.2アンインストール

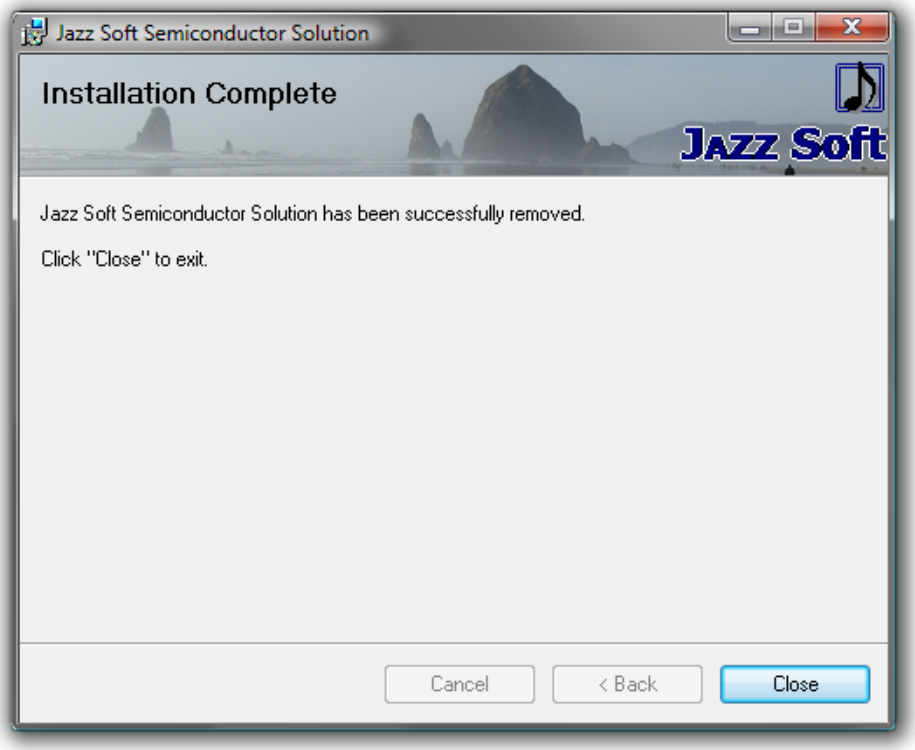
アンインストールはコントロールパネルの「Programs and Features」から行えます。また Setup.exe を実行しても行えます。

1 インストールしたときの Setup.exe を実行します。修復インストールするか、アンインストールするか聞いてきますので、「Remove Jazz Soft Semiconductor Solution」を選択し、Finish ボタンをクリックします。



2 アンインストールが始まると進捗状況が表示されます。

3 アンインストールが完了しました。Close ボタンをクリックします。



5　チュートリアル

5.1　Visual Basic 2008

ここでは HSMS のミニホストを作成しながらプログラミングについて解説していくことにします。

5.1.1　ミニホストの仕様

作成するミニホストの仕様は以下のようなものとし、通信相手は一般的なウェハ検査装置とします。

- 装置に対してオンライン移行、レシピ指定、測定開始、データ受信を行うことができます。
- 送信できるメッセージは以下のもののみとします。

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

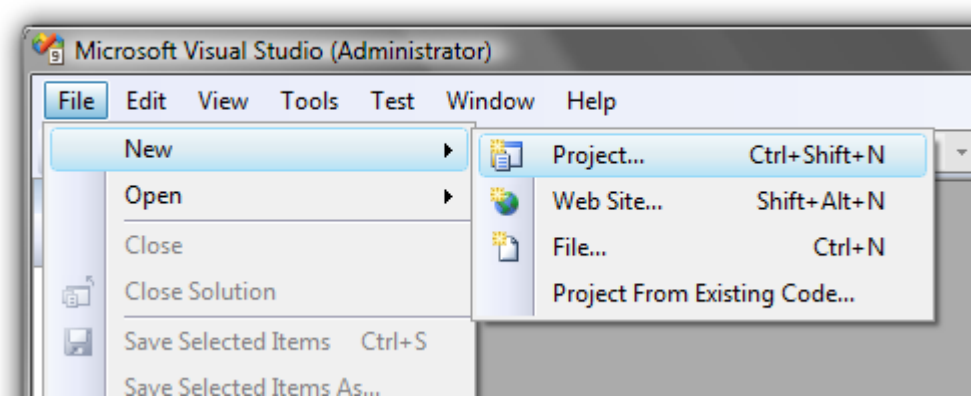
- 受信できるメッセージは以下のもののみとします。

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

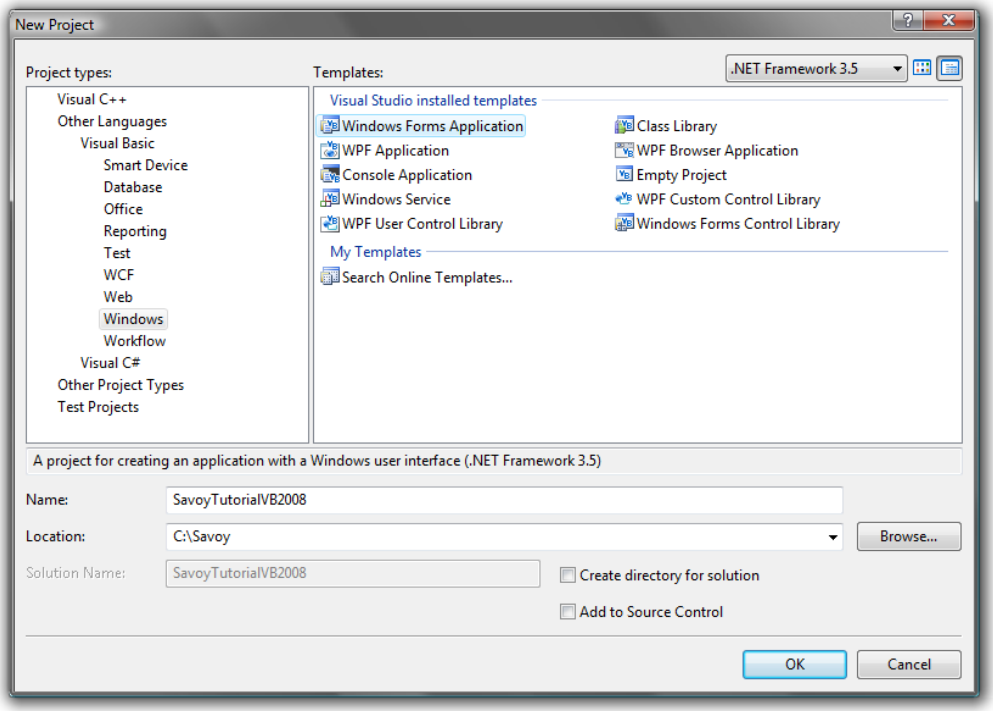
- 装置の初期設定はすでに設定されているものとします。
- ストリーム 9 やファンクション 0 の処理は行わないこととします。
- T3 タイムアウトの監視は行わないこととします。

5.1.2　プロジェクトの作成

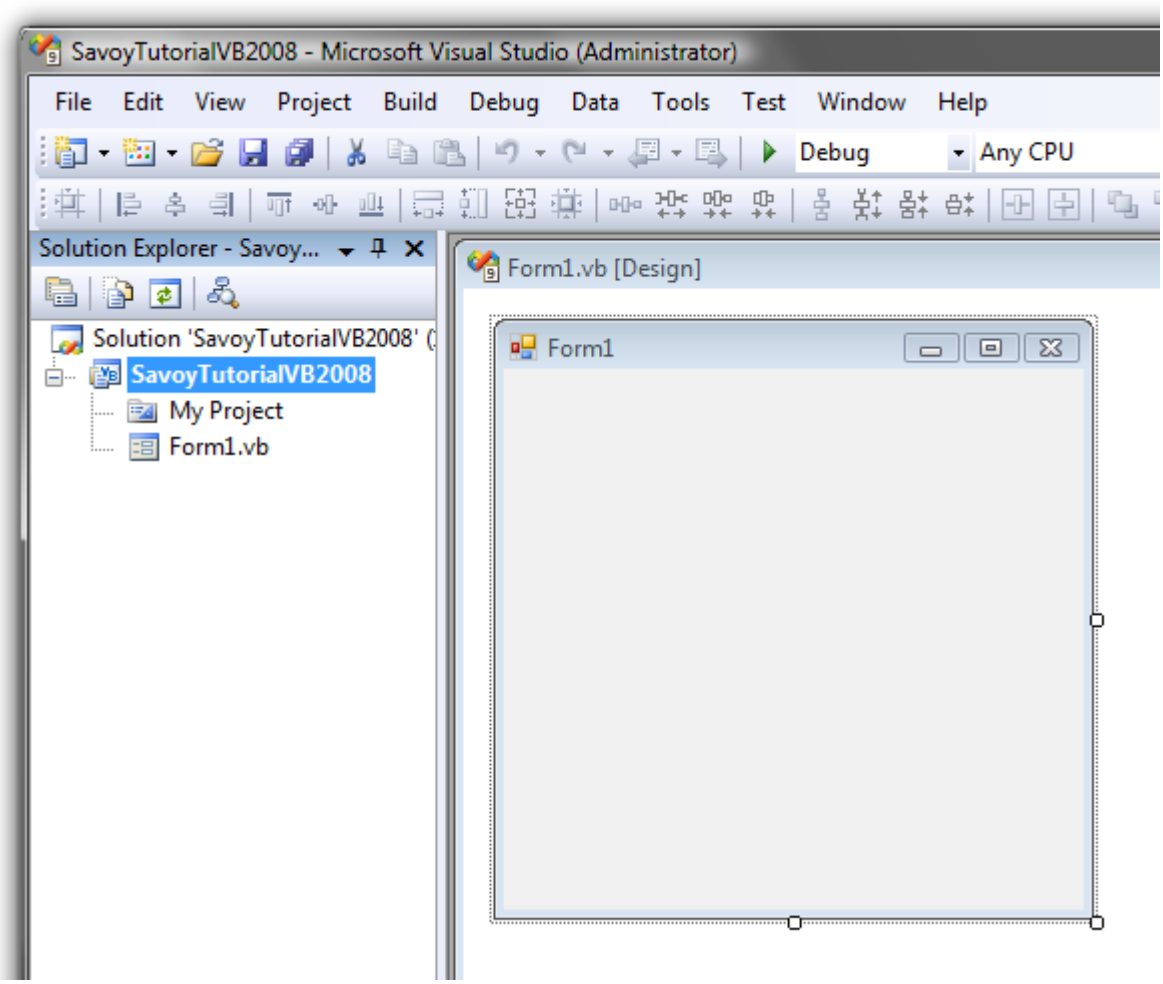
1 Visual Studio 2008 を起動し、File メニューから New – Project…をクリックします。



2 Visual Basic の Windows Forms Application を選択し、プロジェクト名とフォルダを指定します。ここではプロジェクト名を SavoyTutorialVB2008 とします。入力が完了したら OK ボタンをクリックします。



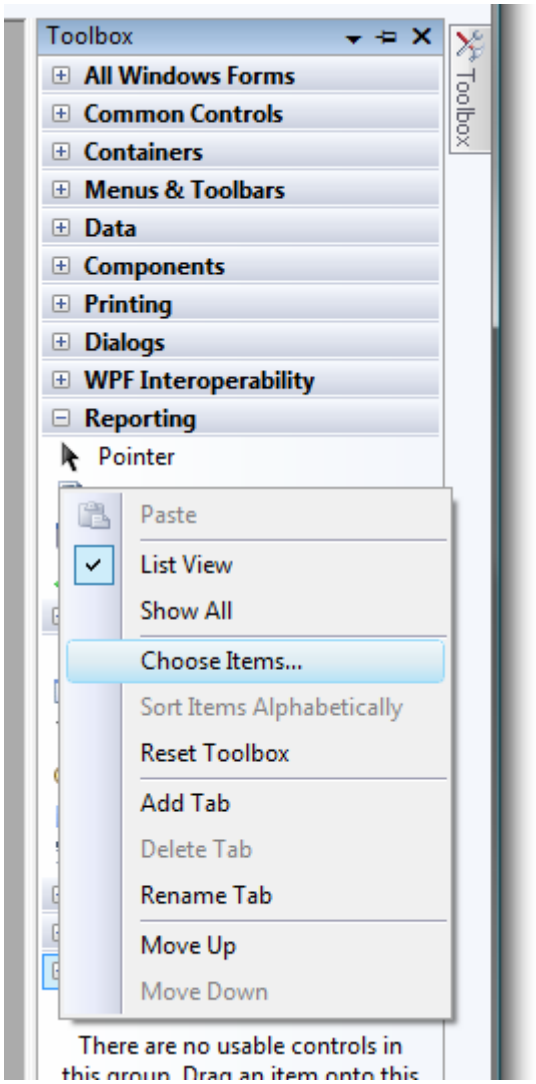
3 新規のプロジェクトが作成されました。



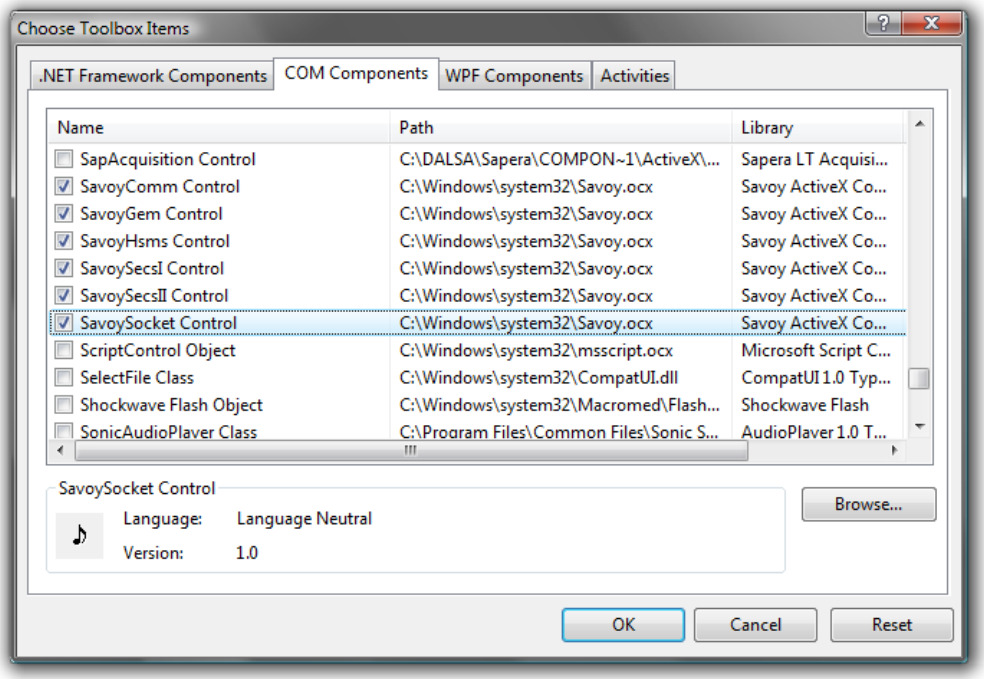
5.1.3　ツールボックスへ Savoy を追加

この作業は 1 回だけ行えば、次回からは不要です。

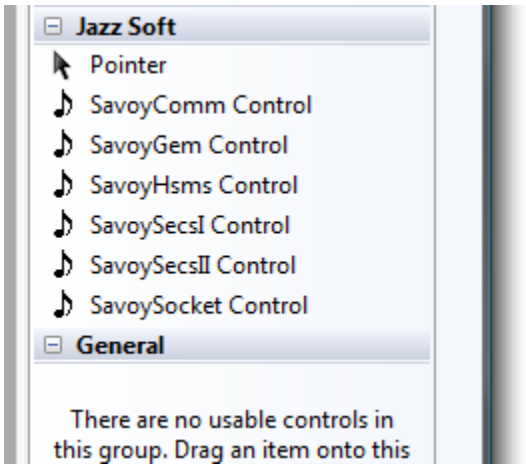
1 Toolbox の余白部分を右クリックし、メニューから Choose Items を選んでクリックします。次の画面が表示されるまでに、1 分以上かかる場合があります。



2 COM Components タブを選択し、一覧から Savoy ActiveX Control module にチェックマークをつけます。そして OK ボタンをクリックします。

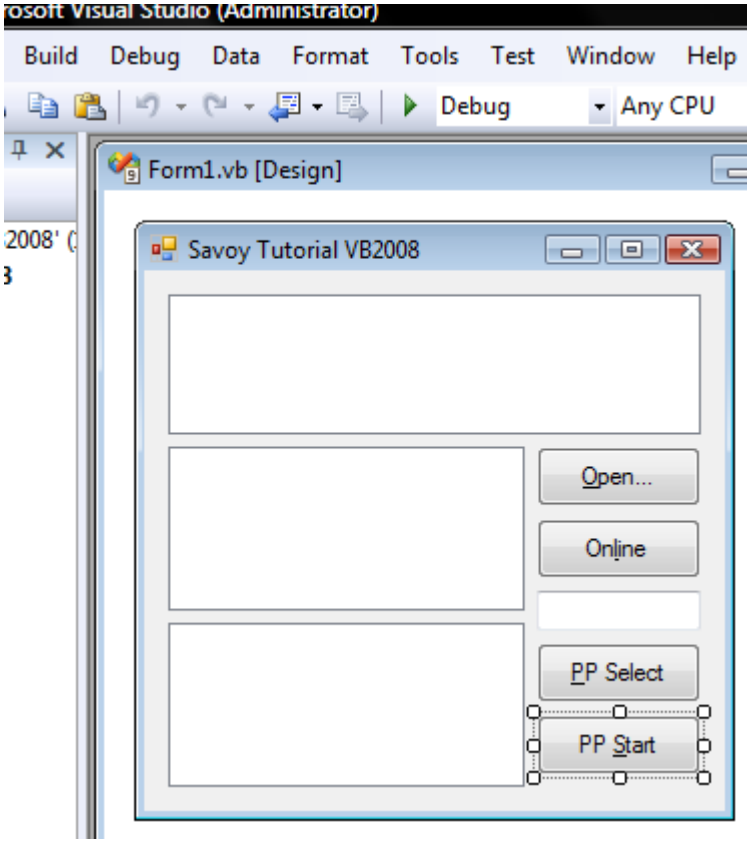


3 Savoy がツールボックスに追加されますので、将来探しやすくなるよう Tab を追加して整理しておきます。



5.1.4 フォームに Savoy を配置

4 下記のように SavoyHsms、SavoySecsII をフォームに配置します。



5 Open ボタンをクリックしたら、SavoyHsms の通信設定画面が表示されるようにします。その後 OK ボタンが押されたら接続するようにします。設定内容は Savoy.ini ファイルに保存され、次回から設定の入力は不要です。

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
```

6 Online ボタンをクリックしたら、S1F13 を送信するようにします。

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
```

7 PP Select ボタンをクリックしたら、リモートコマンドで PP-SELECT を送信するようにします。

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
```

8 PP Start ボタンをクリックしたら、リモートコマンドで START を送信するようにします。

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 イベントの処理

SavoyHsms からのイベントを処理します。

1 Connected イベントが来たら、セレクト要求を送信するようにします。

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 Received イベントが来たら、メッセージ内容を解析するために SavoySecsII に渡します。

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 返信の必要なデータメッセージを受け取ったら、適当な返事を返信します。

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 セレクト要求を受け取ったら、セレクト応答を返信します。

Visual Basic 2008

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

5.1.6 全ソースコード

以上でミニホストの完成です。このプロジェクトはゼロからスクラッチで作った訳ですが、空白行やコメントを入れてもたったの 55 行しかありません。実際に自分で書いたコードの行数は、コメントを除くとなんと 25 行です。他社製品にあるような訳の分からない設定ファイルもデータファイルも書きませんでした。このような驚異的なまでの簡単さは、他社では真似のできない芸当です。

Visual Basic 2008

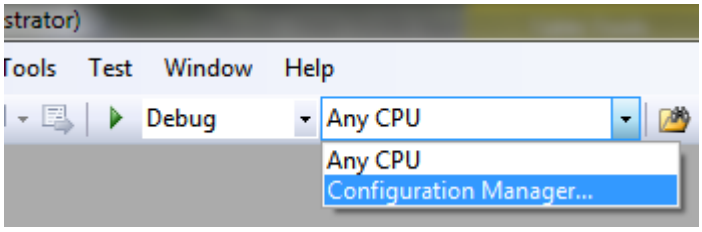
```
Public Class Form1

    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
```

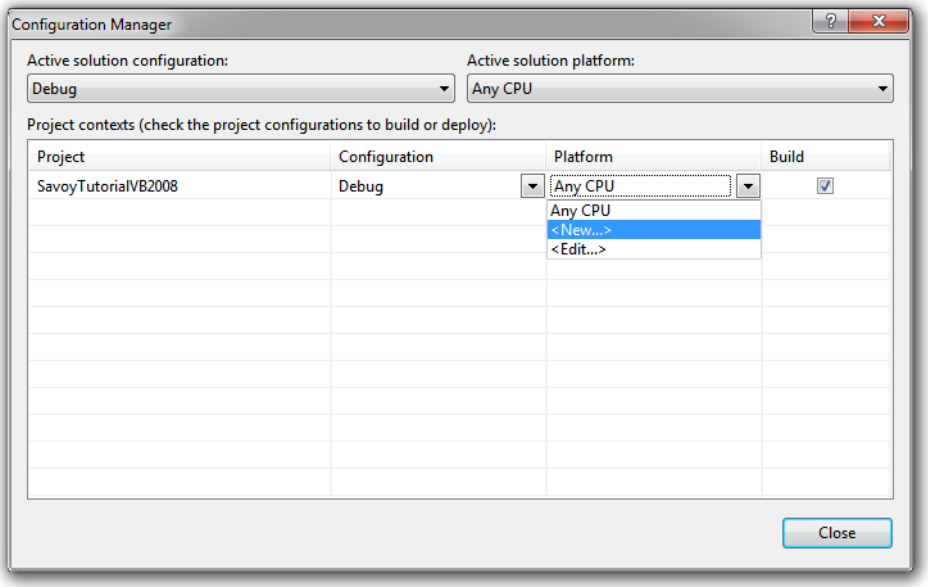
5.1.7 Windows 64 ビット版における注意点

Windows の 64 ビット版で動作させるにはアプリケーションを x86 モードにする必要があります。以下のようにプロジェクト設定を変更してください。

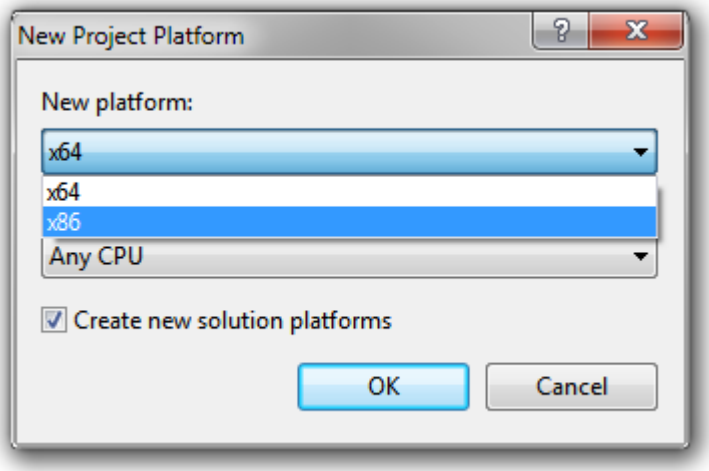
1 ツールバーから Configuration Manager をクリックします。



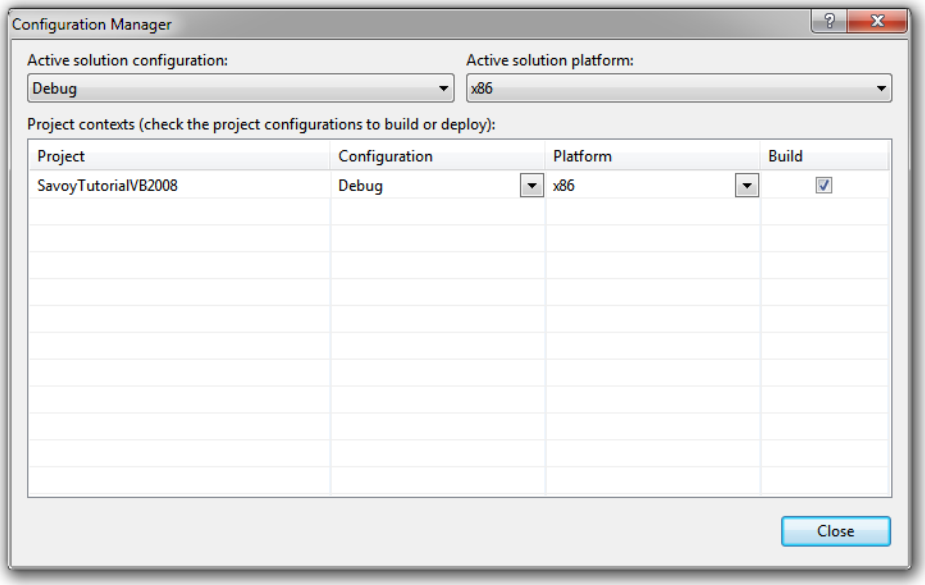
2 Platform から<New...>をクリックします。



3 New platform から x86 を選択し、OK ボタンをクリックます。



4 Platform が x86 に変更されていることを確認します。



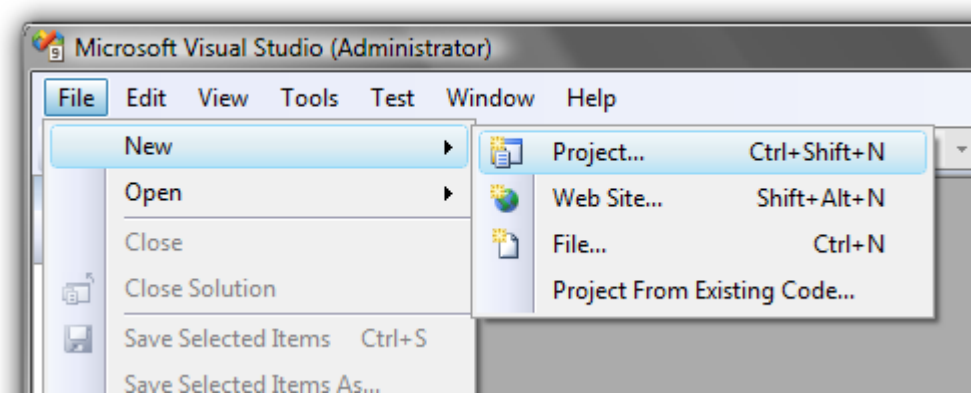
5 上記は Debug 版ですが、Release 版も同様に変更します。

5.2 C# 2008

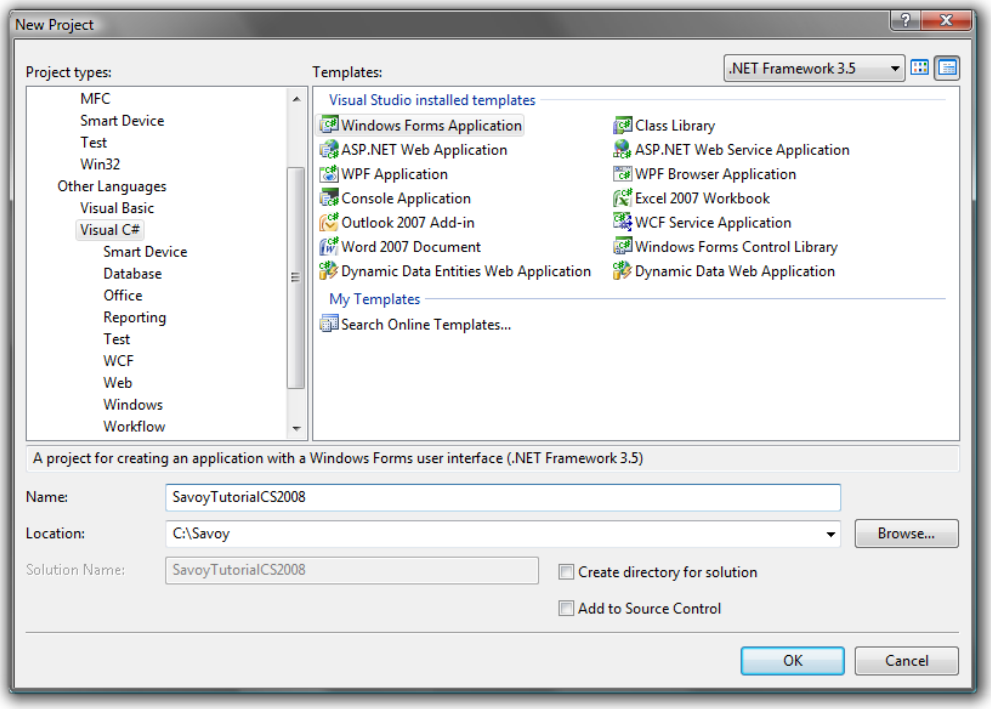
先ほどの Visual Basic 2008 で作成したアプリケーションを、今度は C# 2008 で書いてみましょう。

5.2.1 プロジェクトの作成

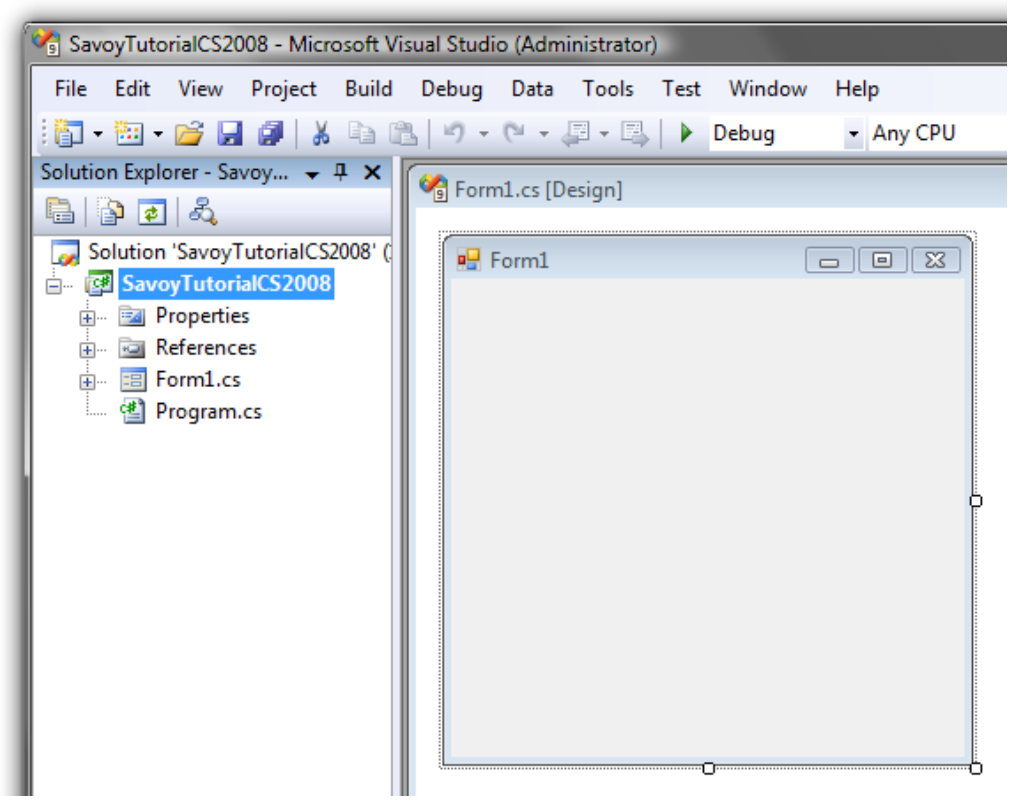
1 Visual Studio 2008 を起動し、File メニューから New – Project…をクリックします。



2 Visual C#の Windows Forms Application を選択し、プロジェクト名とフォルダを指定します。ここではプロジェクト名を SavoyTutorialCS2008 とします。入力が完了したら OK ボタンをクリックします。

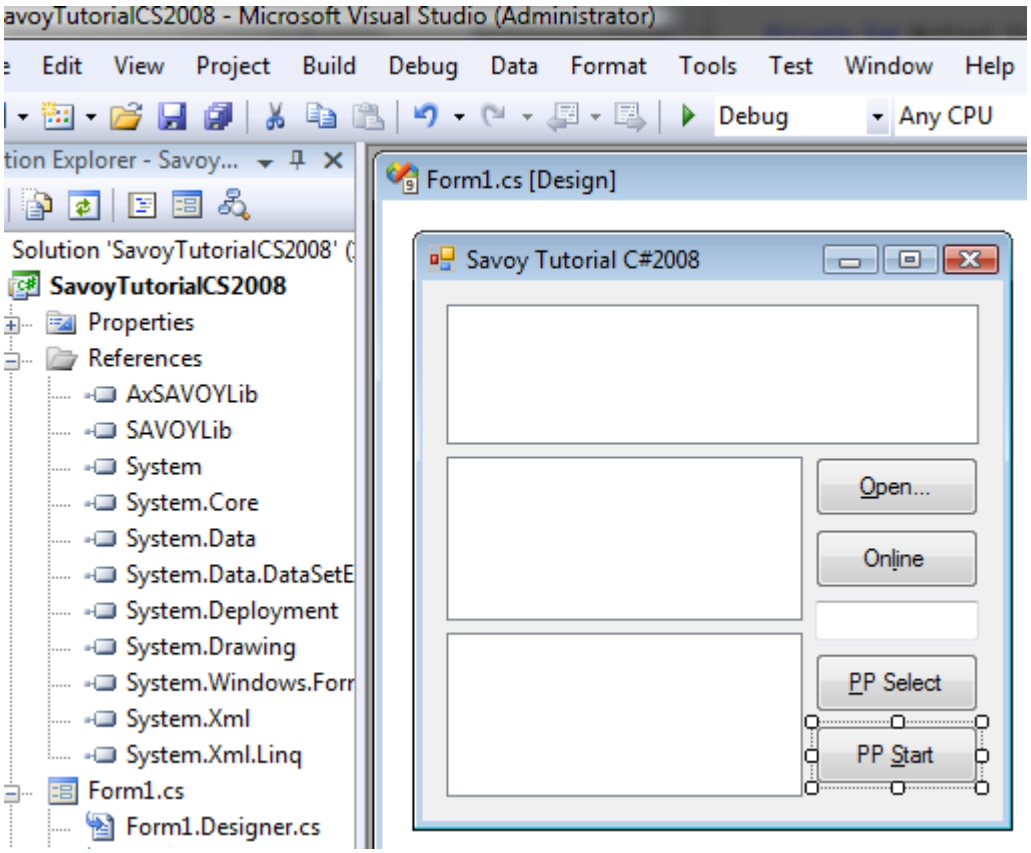


3 新規のプロジェクトが作成されました。



5.2.2　フォームに Savoy を配置

1　下記のように SavoyHsms、SavoySecsII をフォームに配置します。



2 Open ボタンをクリックしたら、SavoyHsms の通信設定画面が表示されるようにします。その後 OK ボタンが押されたら接続するようにします。設定内容は Savoy.ini ファイルに保存され、次回から設定の入力は不要です。

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 Online ボタンをクリックしたら、S1F13 を送信するようにします。

Visual Basic 2008

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

4 PP Select ボタンをクリックしたら、リモートコマンドで PP-SELECT を送信するようにします。

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 PP Start ボタンをクリックしたら、リモートコマンドで START を送信するようにします。

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 イベントの処理

SavoyHsms からのイベントを処理します。

1 Connected イベントが来たら、セレクト要求を送信するようにします。

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 Received イベントが来たら、メッセージ内容を解析するために SavoySecsII に渡します。

Visual Basic 2008

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

3 返信の必要なデータメッセージを受け取ったら、適当な返事を返信します。

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 セレクト要求を受け取ったら、セレクト応答を返信します。

Visual Basic 2008

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

5.2.4 全ソースコード

以上でミニホストの完成です。このプロジェクトはゼロからスクラッチで作った訳ですが、空白行やコメントを入れてもたったの 83 行しかありません。実際に自分で書いたコードの行数は、コメントを除くとなんと 30 行です。他社製品にあるような訳の分からない設定ファイルもデータファイルも書きませんでした。このような驚異的なまでの簡単さは、他社では真似のできない芸当です。

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 Windows 64 ビット版における注意点

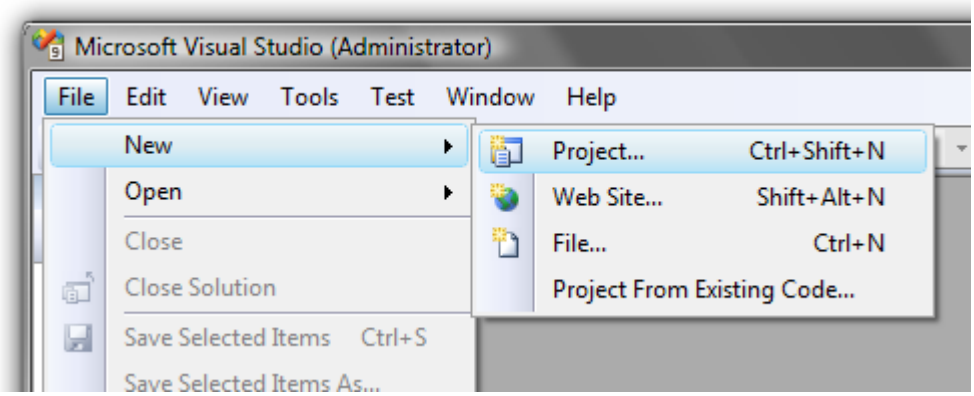
Windows の 64 ビット版で動作させるにはアプリケーションを x86 モードにする必要があります。上記 Visual Basic 2008 を参考にプロジェクト設定を変更してください。

5.3 Visual C++ 2008

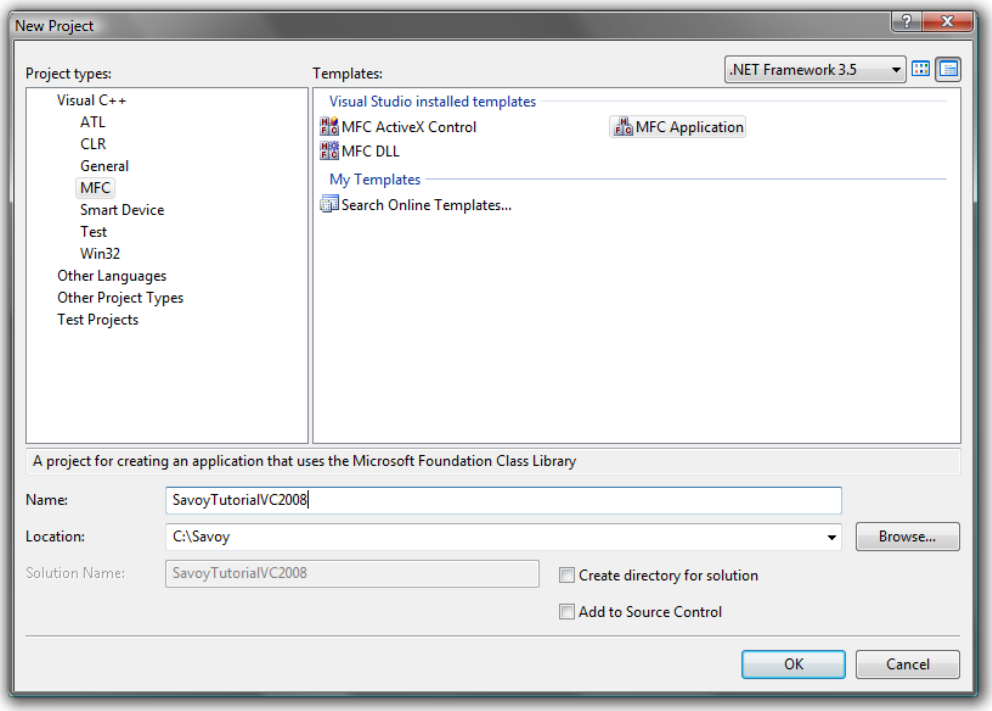
Visual Basic 2008 と C# 2008 では、作成手順やソースコードは極めて似ていましたが、VC++ 2008 の場合は少し異なります。

5.3.1 プロジェクトの作成

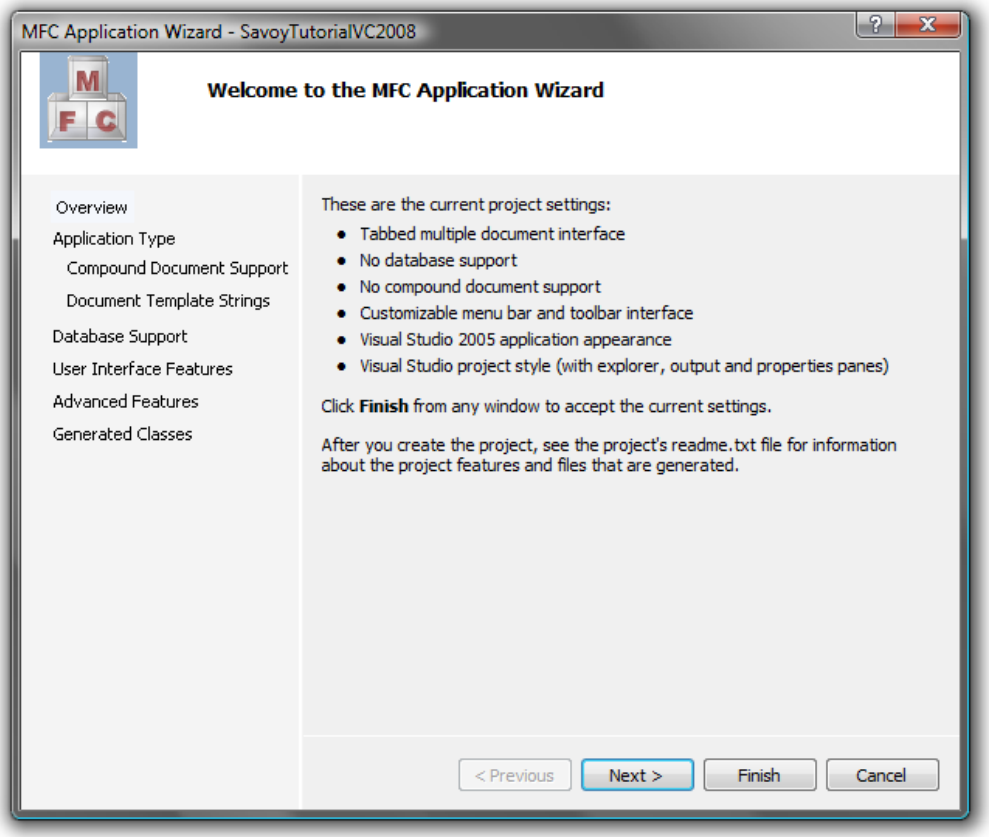
1 Visual Studio 2008 を起動し、File メニューから New – Project…をクリックします。



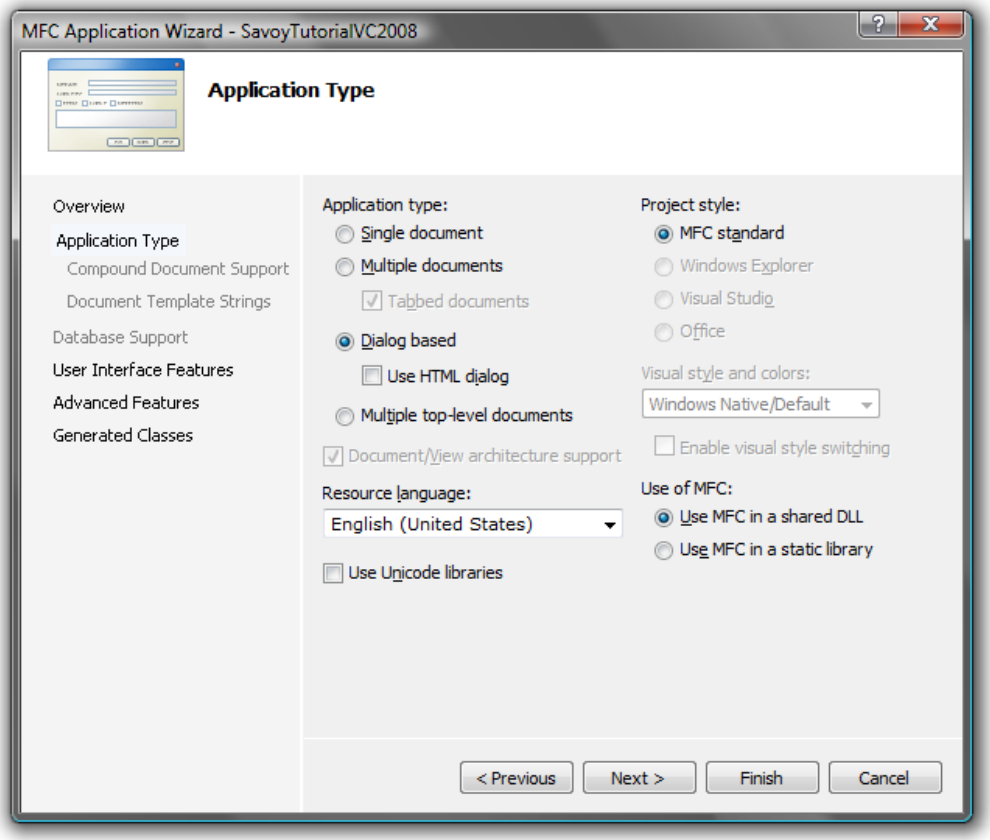
2 Visual C++の MFC Application を選択し、プロジェクト名とフォルダを指定します。ここではプロジェクト名を SavoyTutorialVC2008 とします。入力が完了したら OK ボタンをクリックします。



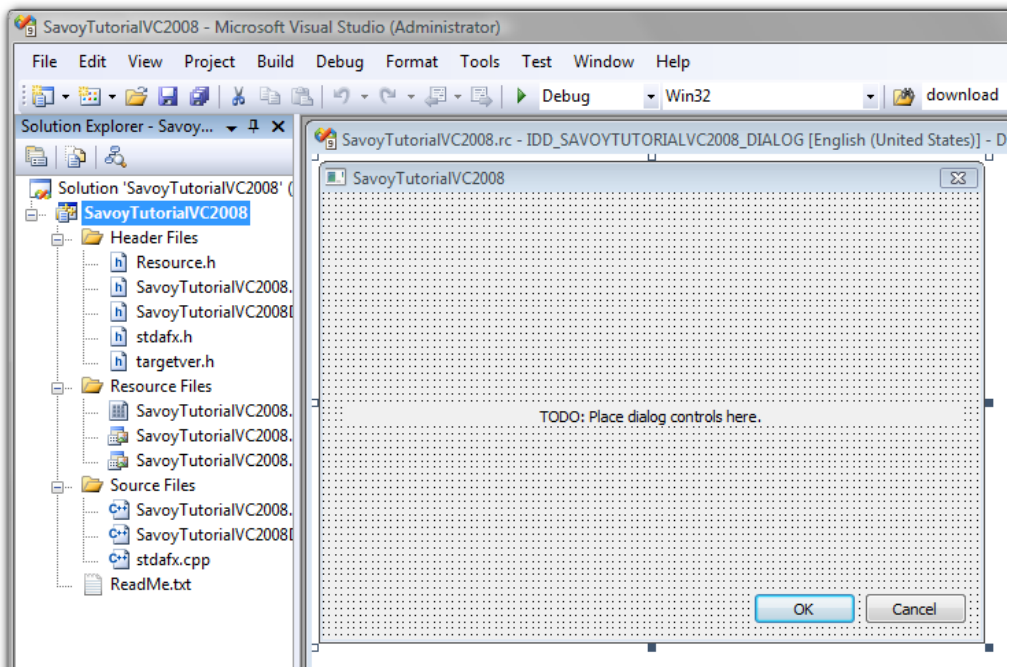
3 ウェルカム画面が表示されます。Next ボタンをクリックします。



4 Dialog based を選択し、Use Unicode libraries のチェックははずします。他の設定はデフォルトのままで構わないので、Finish ボタンをクリックします。

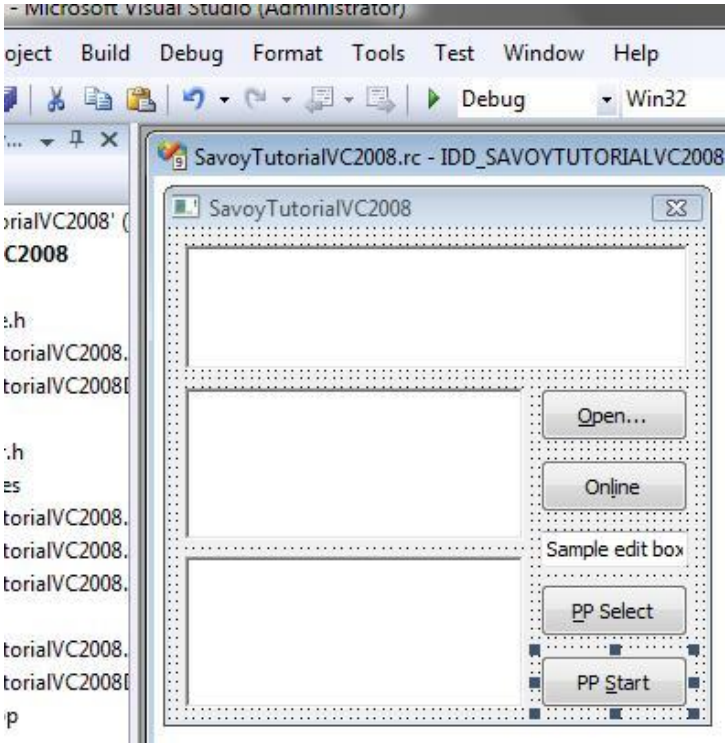


5 新規のプロジェクトが作成されました。

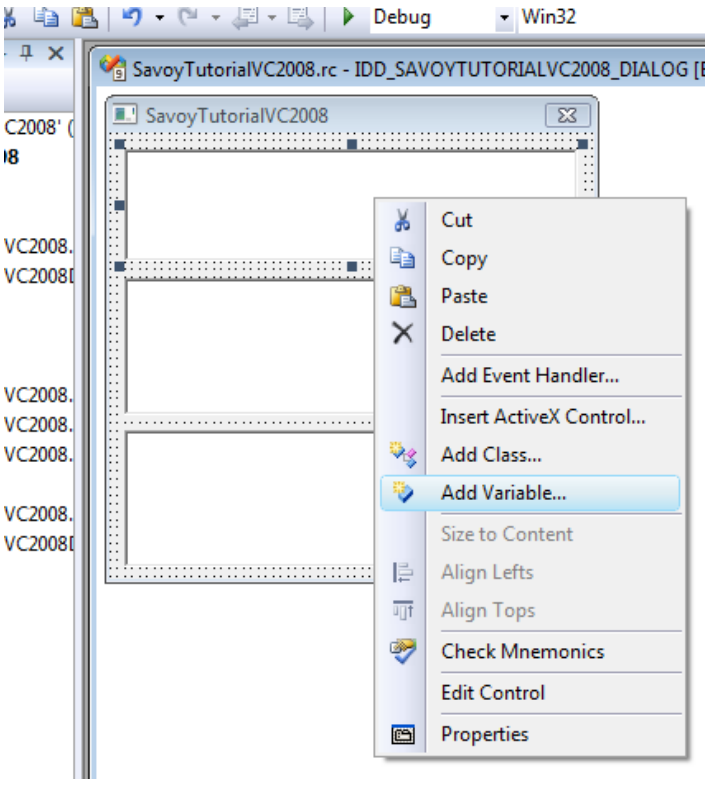


5.3.2 フォームに Savoy を配置

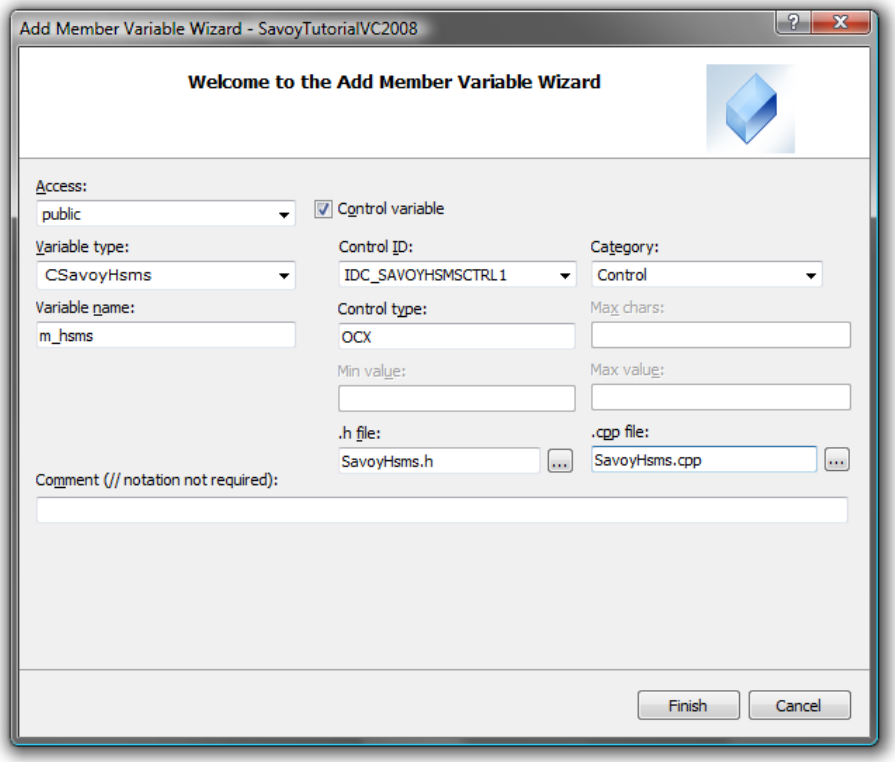
1 下記のように SavoyHsms、SavoySecsII をダイアログリソースに配置します。



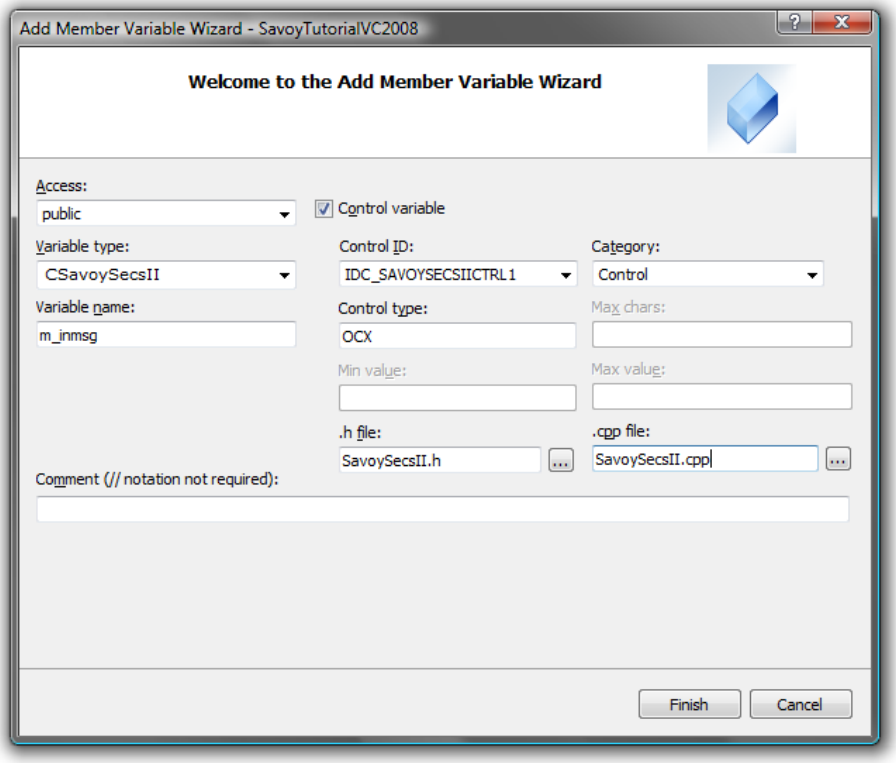
2 変数として割り当てます。SavoyHsms オブジェクトを選択して右クリックし、Add Variable をクリックします。



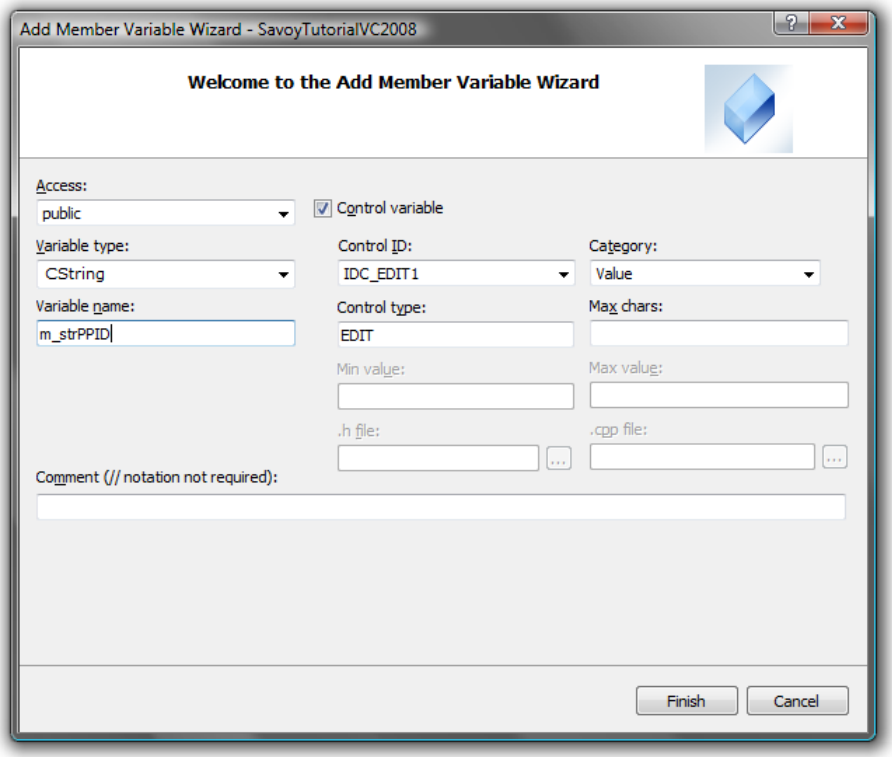
3 Variable type を CSavoyHsms に変更し、.h file と.cpp file もそれぞれ SavoyHsms.h、SavoyHsms.cpp とします。Variable name は m_hsms とし、Finish ボタンをクリックします。



4 同様に SavoySecsII にも変数を割り当てます。Variable type を CSavoySecsII に変更し、.h file と.cpp file もそれぞれ SavoySecsII.h、SavoySecsII.cpp とします。Variable name は m_inmsg および m_outmsg とします。



5 レシピ名は CString 型の変数とし、m_strPPID とします。



5.3.3 ラッパークラスの置き換え

Visual C++ 2008 は ActiveX コントロールのラッパークラス作成に関する、いくつかの問題があります。このため Jazz Soft では不具合対策用のラッパークラスを提供しています。ラッパークラスは Visual C++ 6.0 でも作成可能ですが、enum が反映されません。

1 Visual C++ 2008 を終了し、プロジェクトをいったん閉じます。

2 SavoyHsms.h、SavoyHsms.cpp、SavoySecsII.h、SavoySecsII.cpp を上書きします。

3 再び Visual C++ 2008 を起動し、プロジェクトを読み込み直します。

5.3.4 ボタンの処理

1 Open ボタンをクリックしたら、SavoyHsms の通信設定画面が表示されるようにします。その後 OK ボタンが押されたら接続するようにします。設定内容は Savoy.ini ファイルに保存され、次回から設定の入力は不要です。

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 Online ボタンをクリックしたら、S1F13 を送信するようにします。

Visual Basic 2008

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

3 PP Select ボタンをクリックしたら、リモートコマンドで PP-SELECT を送信するようにします。

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());
}
```

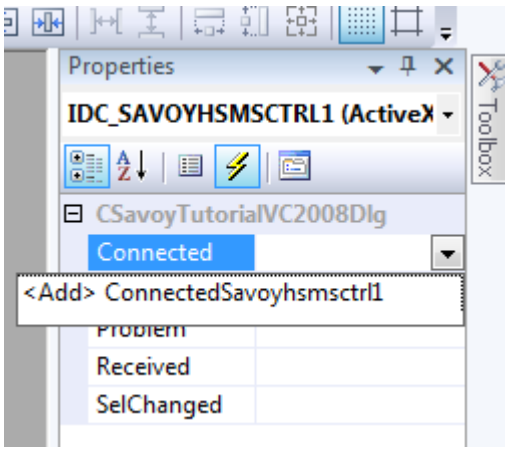
4 PP Start ボタンをクリックしたら、リモートコマンドで START を送信するようにします。

Visual Basic 2008

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

5.3.5 イベントの処理

SavoyHsms からのイベントを処理します。イベントハンドラは、プロパティウィンドゥから作成できます。



1 Connected イベントが来たら、セレクト要求を送信するようにします。

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 Received イベントが来たら、メッセージ内容を解析するために SavoySecsII に渡します。

Visual Basic 2008

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

3 返信の必要なデータメッセージを受け取ったら、適当な返事を返信します。

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 セレクト要求を受け取ったら、セレクト応答を返信します。

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 全ソースコード

以上でミニホストの完成です。全ソースコードはいくつかのファイルで構成されますので、ここでは載せないことにします。

このプロジェクトはゼロからスクラッチで作った訳ですが、ソフトの中心である SavoyTutorialVC2008Dlg.cpp ファイルは、空白行やコメントを入れてもたったの 231 行しかありません。実際に自分で書いたコードの行数は、コメントを除くとなんと 31 行です。他社製品にあるような訳の分からない設定ファイルもデータファイルも書きませんでした。このような驚異的なまでの簡単さは、他社では真似のできない芸当です。