



---

**Windows Tutorial for  
ASNA Visual RPG 9.0 for Microsoft Visual Studio 2008**



## **Windows Tutorial for Visual RPG 9.0 for Microsoft Visual Studio 2008**

Information in this document is subject to change without notice. Names and data used in examples are fictitious unless otherwise noted.

No component of **Windows Tutorial for Visual RPG 9.0 for Visual Studio 2008** may be reproduced, disassembled, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form without the written permission of ASNA (Amalgamated Software of North America).

Copyright ©2003 – 2008 ASNA - Amalgamated Software of North America. All rights reserved.

**Release 9.0 - August 4, 2008**

---

# CONTENTS

<b>Introduction .....</b>	<b>1</b>
Using the Tutorial .....	2
Source Files .....	2
Manual Conventions .....	3
<b>Step 1: Installing Necessary Components .....</b>	<b>5</b>
Installing Visual RPG for Visual Studio 2008 .....	6
Installation for New Users .....	6
Licensing Visual RPG for Visual Studio 2008 .....	7
<b>Step 2: Creating the Windows Form .....</b>	<b>9</b>
Creating the Windows Project .....	10
Examining Windows Form Structure .....	11
Creating the Windows Form – (Adding Controls and Text) .....	12
Running the Windows Application .....	24
Step 2 Summary .....	25
<b>Step 3: Responding to Events .....</b>	<b>27</b>
Adding Code to Form1.vr .....	28
Step 3 Summary .....	31
<b>Step 4: Database Access .....</b>	<b>33</b>
Adding Code to Form1 – Accessing a Database File .....	34
Declaring Database and File Objects .....	35
Connecting to a Database .....	36
Opening a File .....	36
Retrieving Customer Information (btnSearch_Click) .....	36
Closing a Database and File (btnExit_Click) .....	37
Step 4 Summary .....	39
<b>Additional Exercises .....</b>	<b>40</b>

<b>Appendix A: Visual Studio 2008 101.....</b>	<b>41</b>
Forms Designer .....	41
Solution Explorer .....	41
Properties .....	43
Toolbox .....	44
Tools - Options.....	45
Tools – Options – Environment - General .....	45
Environment - Fonts and Colors .....	46
CMastNewL1 .....	47
<b>Index .....</b>	<b>49</b>

# Introduction to Visual RPG Windows Tutorial for Visual Studio 2008

This tutorial will guide you through the steps necessary to write a basic windows application using Visual RPG for Visual Studio 2008.

After completing this tutorial, you will have a working windows application written in Visual RPG that will allow you to connect to a database, retrieve a customer file record, and display the customer details.

---

**At the end of this tutorial, the windows application will look like the following:**

**CUSTOMER INQUIRY**

**Customer Inquiry**

Enter Customer Number:

Name:

Address:

City:

State:

Post Code:

Phone:

Fax:

## Getting Started

This tutorial does not assume that you have any prior experience with Visual RPG or with creating Windows applications. However, due to the extent of information for Visual Studio 2008 and Visual RPG for Visual Studio 2008, you will still want to refer to the on-line documentation for further assistance.

## Using the Tutorial

This tutorial is designed as a “self-learning” tool and includes 4 steps, or chapters. Each chapter is a “step” in which a portion of the application is created. Each step will build upon the previous step, so it is recommended that you start at Step 1 and progressively work towards the end. Each step should take you approximately an hour or less. An approximate time to complete each step is included at the beginning of each chapter.

### **At the beginning of each chapter, you will find:**

- A summary of the topics covered in that step.
- An approximate time to complete that step.
- What the application will look like at the completion of that step.

### **At the end of each chapter, you will find:**

- A reference section of each task performed in that step, including how to perform that task, along with the button or shortcut key that is used to perform that task (if any).

### **At the end of this tutorial, you will find:**

- **Appendix A** – Visual Studio 2008 Tools. This appendix describes some of the Windows or views you have in Visual Studio 2008 and information on getting around in Visual Studio 2008.


*You may want to look at this chapter **First**; especially if you are not familiar with the Visual Studio IDE windows.*

## Source Files

There are no source files accompanying this tutorial (as compared to other ASNA tutorials). There is such a small amount of code for you to enter, so it really was not necessary to include the completed code for you to compare your code to.

## Manual Conventions

The following is a list of conventions that are used throughout this tutorial representing particular functions.

- A  beside a headline indicates a step-by-step process that you are to perform.
- Numbered lists (1, 2, 3, and so on) indicate steps that you should follow.
- Menu options, buttons, controls and keys that you are to select, along with words that you are to type, will appear in **boldface**.
- *Notes and hints are in italicized font, and some with a separator line above and below the note.*
- There are several icons that will appear in the left margin next to the instructions. These icons are a visual representation of a task to perform, a button to select, text to enter, or a Windows form address to enter, such as shown below:



Indicates that you are to select a particular control from the Toolbox (the icon will vary depending upon the control to select).

**This Form Intentionally Left Blank**

# Step 1: Installing Necessary Components

---

## What you will learn in Step 1:

- Installing Visual RPG for Visual Studio 2008 for previous and new users.
- Licensing Visual RPG for Visual Studio 2008.

## Approximate Time to Complete Step 1:

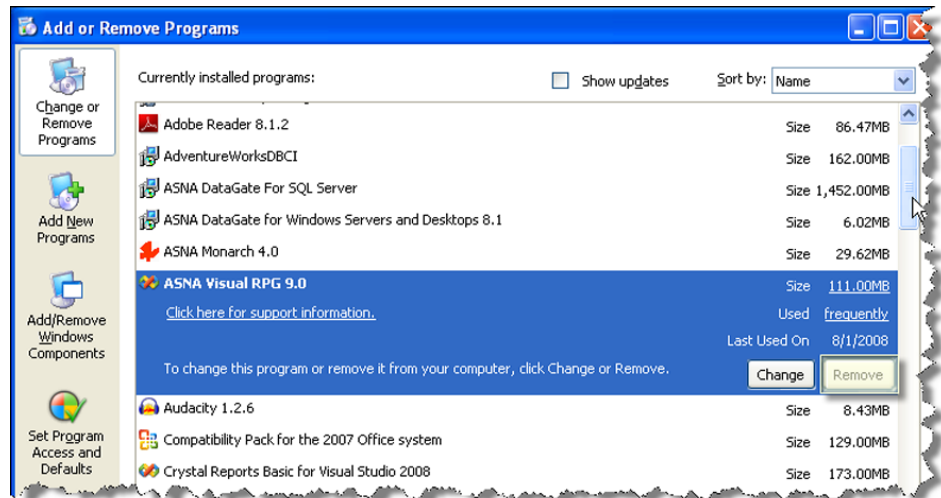
Will vary depending upon the components to install, but could take up to an hour.

## Installing Visual RPG for Visual Studio 2008

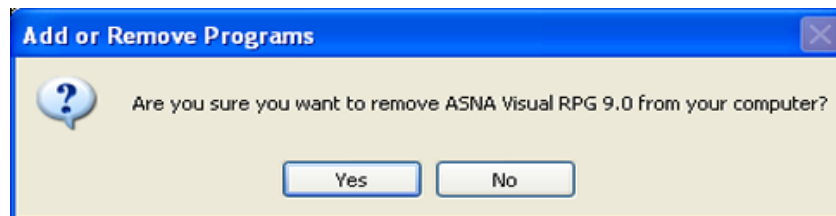
### ✓ Installing Visual RPG for Visual Studio 2008 for previous Users

1. If this is **NOT** your first installation of ASNA Visual RPG 9.0 for Visual Studio 2008, you must first uninstall the previous version by going to the **Start Menu - Control Panel – Add or Remove Programs**.

Your computer will compile a list of programs currently installed on your machine in a window that looks something like this:



2. Click on **ASNA Visual RPG 9.0 for Visual Studio 2008** and select the **Remove** button.
3. In the dialog that displays next, select **Remove** and click **Next>**.
4. You will be prompted with a message asking if you want to completely remove the selected application. Select **Yes** to remove Visual RPG 9.0, or select **No** to cancel.



5. Once the older version of the product is uninstalled, you may continue to the next section and install the current build as if you are a new user.

## Installation for New Users

### ✓ Installing Visual RPG 9.0 for Visual Studio 2008 for New Users

If this is your first time installing Visual RPG for Visual Studio 2008 (i.e. you do not have any previous version on your machine), then installation is very straightforward. If you encounter difficulties during the installation process, refer to the [Visual RPG Installation Guide](#) for more details.

1. Simply run the **Setup** file, and accept all the default settings (this equates to basically clicking **Next** a bunch of times). Once this is complete, you are ready to **license** your copy of Visual RPG for Visual Studio 2008 as per the following steps.


## Licensing Visual RPG for Visual Studio 2008

To complete this tutorial, you must have a valid license of Visual RPG 9.0. Obtain your license key from ASNA. Enter the license key using ASNA's registration and licensing tool called **ASNA Registration Assistant**.

Follow the steps below to license the Visual RPG 9.0 for Visual Studio 2008.

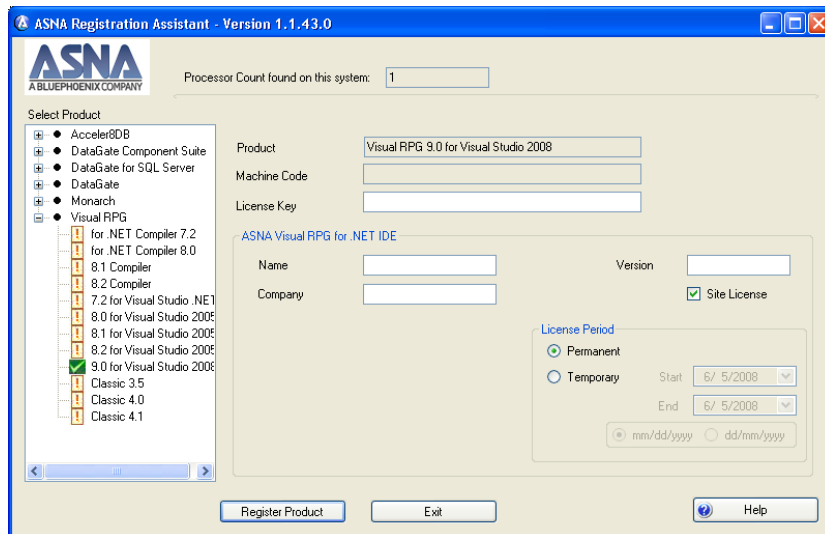
If you have never licensed the Visual RPG for Visual Studio 2008 product suite, then you must follow these steps in order to use the product. ***Bear in mind that many of the beta builds did not require licenses to run, so just because you have been using a beta version does not mean that you are automatically registered.***

### To License Visual RPG 9.0 for Visual Studio 2008

1. Contact **ASNA** to obtain a License Key.
2. Start ASNA Registration Assistant by selecting **Start - ASNA Product Suite for Visual Studio 2008 -  ASNA Registration Assistant**.

*You can also select **Asna Registration Assistant.exe** from the `\Program Files\Common Files\Asna Shared` folder.*

3. The Registration Assistant dialog will display. You must first select the Product that you are licensing. Select **Visual RPG - 9.0 for Visual Studio 2008**.
4. Enter the **License Key** that you received from ASNA, along with the other pertinent information, such as Name, Company, and License Period.
5. Select the **Register Product** button to accept the information, the **Exit** button to close Registration Assistant, or the **Help** button to get help on ASNA Registration Assistant.



ASNA Registration Assistant - Version 1.1.43.0

Processor Count found on this system: 1

Select Product

- Acceler8DB
- DataGate Component Suite
- DataGate for SQL Server
- DataGate
- Monarch
- Visual RPG
  - for .NET Compiler 7.2
  - for .NET Compiler 8.0
  - 8.1 Compiler
  - 8.2 Compiler
  - 7.2 for Visual Studio .NET
  - 8.0 for Visual Studio 2005
  - 8.1 for Visual Studio 2005
  - 8.2 for Visual Studio 2005
  - 9.0 for Visual Studio 2008
  - Classic 3.5
  - Classic 4.0
  - Classic 4.1

Product: Visual RPG 9.0 for Visual Studio 2008

Machine Code:

License Key:

ASNA Visual RPG for .NET IDE

Name:  Version:

Company:   Site License

License Period

Permanent  Temporary

Start: 6/ 5/2008 End: 6/ 5/2008

mm/dd/yyyy  dd/mm/yyyy

Register Product Exit Help

**Congratulations!**

---

**You should have all of your Visual RPG Windows components installed and licensed, so you are now ready to begin creating your Windows application in the next step.**

## Step 2: Creating the Windows Form

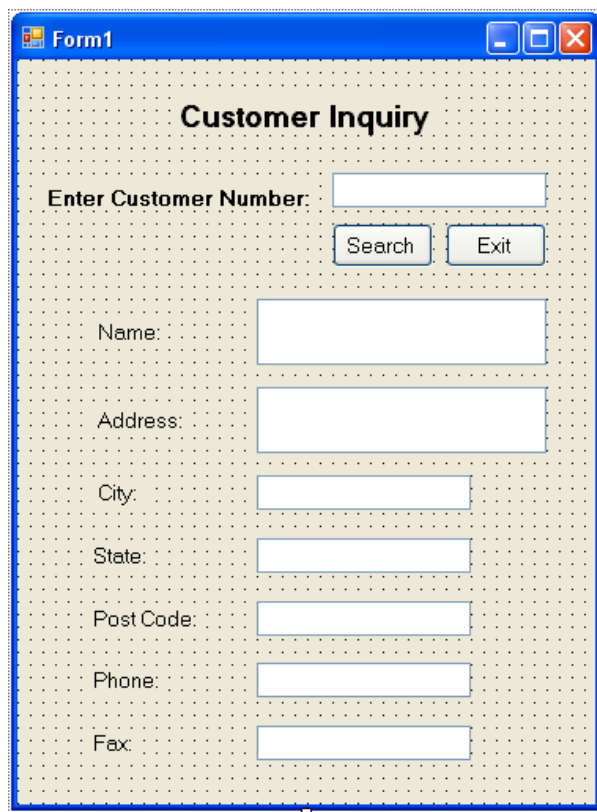
### What you will learn in Step 2:

- How to create a windows project in Visual Studio 2008.
- How to view the Toolbox and add controls to a form.
- How to align a group of controls.
- How to run your windows application.

### Approximate Time to Complete Step 2:

1 hour.

### What the Windows Form will look like after completing Step 2:



The screenshot shows a Windows application window titled "Form1" with a standard Windows XP-style title bar (minimize, maximize, close buttons). The main area of the window has a light gray background with a fine grid pattern. At the top center, the text "Customer Inquiry" is displayed in a bold, black font. Below this, the form contains the following elements:

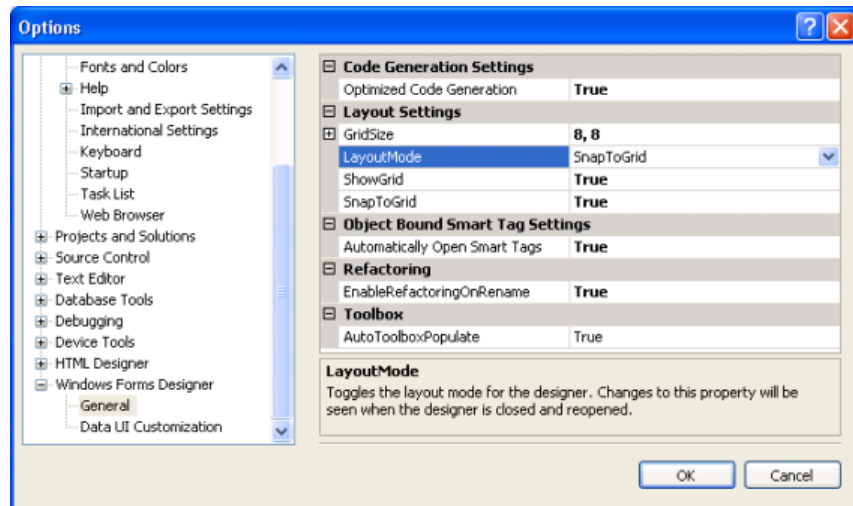
- A label "Enter Customer Number:" followed by a single-line text input field.
- Two buttons, "Search" and "Exit", positioned below the first input field.
- A label "Name:" followed by a single-line text input field.
- A label "Address:" followed by a single-line text input field.
- A label "City:" followed by a single-line text input field.
- A label "State:" followed by a single-line text input field.
- A label "Post Code:" followed by a single-line text input field.
- A label "Phone:" followed by a single-line text input field.
- A label "Fax:" followed by a single-line text input field.

## Creating the Windows Project

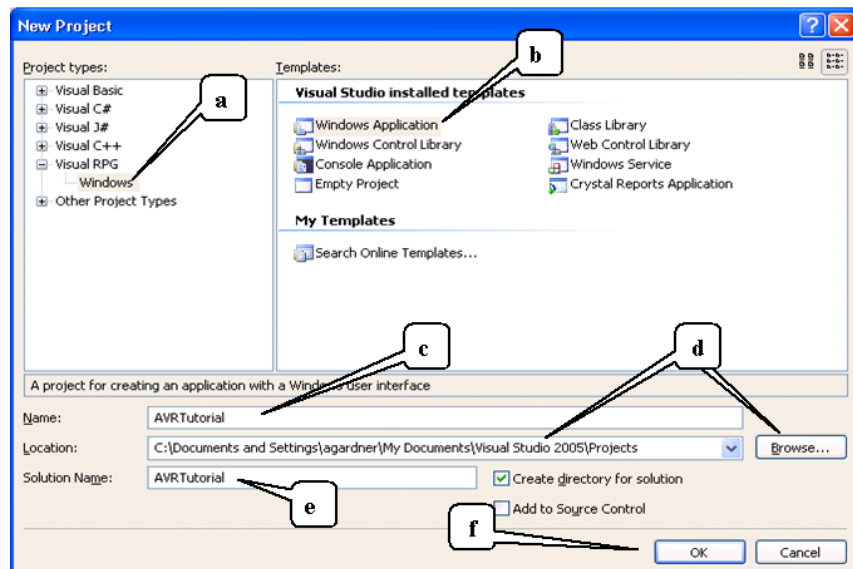
First, we will create a Windows application (which automatically gives you a Windows form).

### ✓ To Create a Windows Project

1. Open **Visual Studio 2008**.
2. In this document, the snap to grid layout mode is used for readability. To set this option, on the **Tools** menu, point to **Options**, then the **Windows Forms Designer General** tab (scroll down to the bottom). Select **SnapToGrid** from the drop down list for the **LayoutMode** property as shown completed below. Click **OK**.



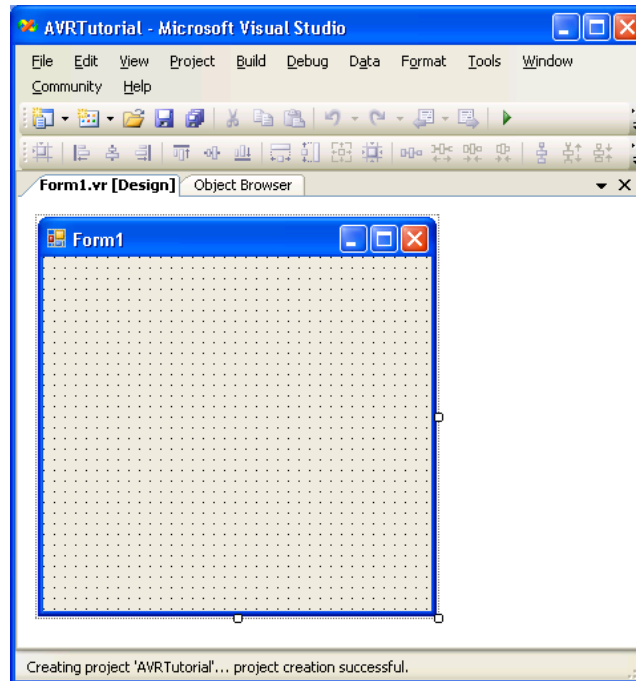
3. On the **File** menu, point to **New**, and then click **Project**, or **Ctrl+Shift+N**.
4. In the **New Project** dialog box, perform the following steps while referring to the dialog below.



- a. In the **Project Types** pane on the left, choose **Visual RPG, Windows**.
- b. In the **Templates** pane on the right, choose **Windows Application**.
- c. Enter the project **Name**, *AVRTutorial* in our example.
- d. We will leave the **Location** set to the default value on your system, but if needed, click on the **Browse** button to navigate to the location of your choice.
- e. We will also leave the **Solution Name** to the default value **AVRTutorial**.
- f. Click **OK**. When you click **OK**, the IDE will automatically perform the necessary steps to set up your AVRTutorial on your system. The AVRTutorial folder is created at the root level of the **Location** folder you selected in the previous step.

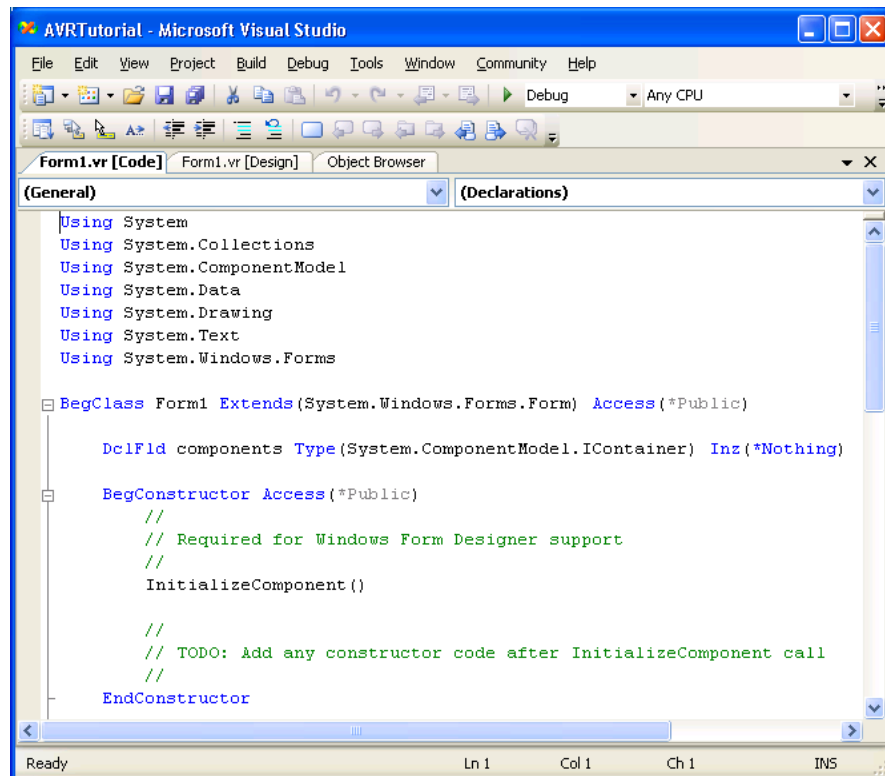
## Examining Windows Form Structure

The Windows Form Designer is opened with a file called **Form1.vr [Design]**. This is the form for which you will begin creating the user interface for the application. Notice the word **[Design]** after **Form1.vr** in the tab. This denotes that you are viewing the form in design mode.



Right-mouse click on the form and choose **View Code** from the context menu that appears. A new window will appear displaying the code for the form.

*Notice the new tab titled **Form1.vr [Code]** to the left of the **Form1.vr [Design]** tab. This denotes that you are viewing the code for the form.*



## Creating the Windows Form – (Adding Controls and Text)

You are now ready to begin creating your first Windows form by adding controls to it.

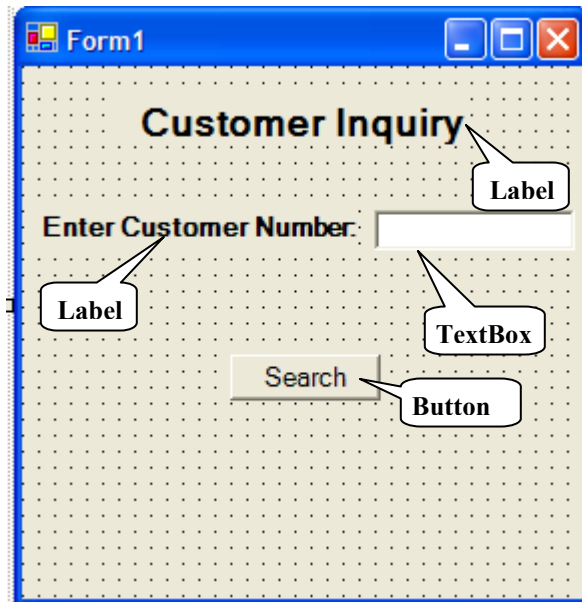
This form will be the user interface where the user enters the customer number and views the customer's information retrieved from the file in the database. To begin, let's place a TextBox onto the Windows form where the user will enter the customer number.


The controls available for use on the Windows form are contained within the **Toolbox** in Visual Studio 2008. The Toolbox displays various **Tabs**; which are divided in categories, and based upon the type of project you are creating, and the References that are added to your project. You can collapse and expand each, add your own custom tabs, etc. For the purposes of this Tutorial, we will be accessing the Controls in the **All Windows Forms** tab.

For more information about the Toolbox, see [Toolbox Overview](#) and [Managing the Toolbox](#) in the Visual Studio help. To access the **Help** – select **Help** – Contents, or **Help** – Index.

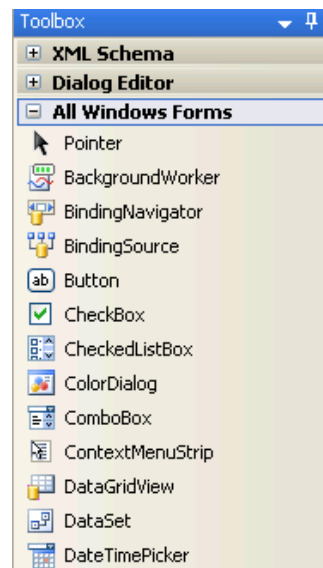
## ✓ Let's Begin Creating the Form

The form that you will initially create will contain the following types of controls:

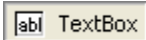


1. Let's go back to the [Design] view of Form1 by clicking on the **Form1.vr[Design]** tab.
2. Display the **Toolbox** containing the controls by selecting **View -  Toolbox**, or by pressing the **Ctrl + Alt + X** keys.

*Notice that the controls available for Windows forms display under the **All Windows Forms** tab. They are also displayed in alphabetical order.*

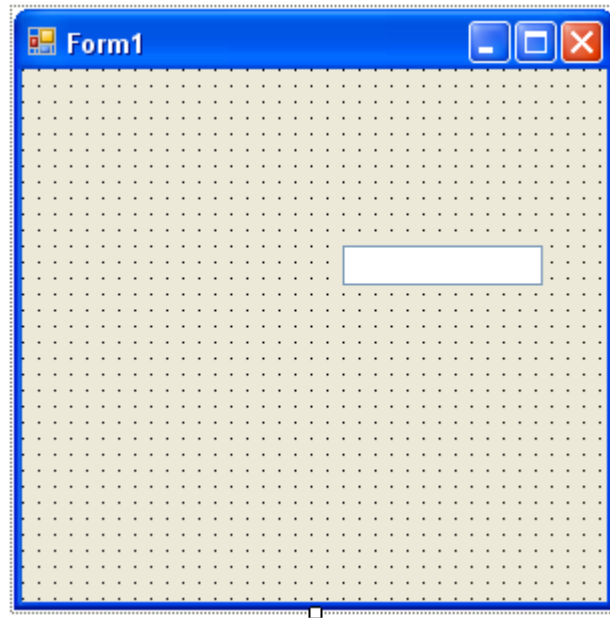


*You may want to click on the other tabs, such as **Data** or **Components** (just to name a few) and notice that the controls available will vary.*



3. Select the **TextBox** control (in the Windows Forms tab) and drag and drop onto the Windows form.

Your Windows form should look something like this:




---

*You can click on the control to **move** it, or you can **resize** the control by grabbing one of the white square "handles" on either side with the mouse pointer.*


---

4. Next, let's change the default name of the TextBox control (textBox1) to a meaningful name that we can recognize in code by changing its **Properties**.

*The Properties Window allows you to view and change the design-time properties and events of selected objects that are located in editors and designers. You can also use the Properties Window to edit and view file, project, and solution properties. The Properties Window is available from the **View** menu.*

*The Properties Window displays different types of editing fields, depending upon the needs of a particular property. These edit fields include edit boxes, drop-down lists, and links to custom editor dialog boxes. Properties shown in gray are read-only. See [Properties Window](#) in the Visual Studio help for more information.*

- a. If the **Properties** Window is not already visible, press **F4** to display it.
- b. Ensure that the textbox you just placed on the form is selected, then locate the field in the Properties Window called **Name**.

*If Properties are presented alphabetically , **Name** will appear near the top of the list with the default name `textBox1`.*

- c. Highlight the name **textBox1** and replace it with **txtCustNo**.

- d. You can stretch the TextBox to your desired length, or you can change the **Size** property in the Properties Window. For this example, I have set the **Size** property to **136, 22**.

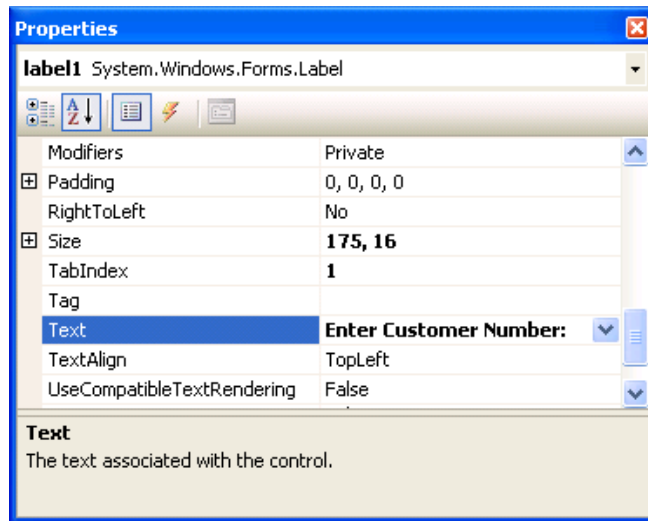
---

*Hint: Start all 'like' controls with the same prefix (e.g. all TextBox names begin with "txt", and all button names begin with "btn"). This makes it easy to see what type of control you are referencing later on when you are coding.*

---



- 5. Let us add a **Label** control next to the TextBox to indicate what this TextBox is asking for.
  - a. Click and drag the **Label** control in the Toolbox onto the form to the left of the TextBox.
  - b. Change its **Text** property from **label1** to **Enter Customer Number:**.

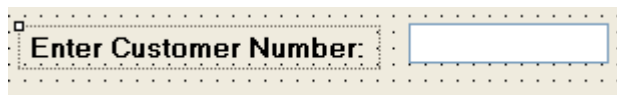


*Changing the (Name) property of the label is not usually necessary since you will probably never need to reference this control in the code.*

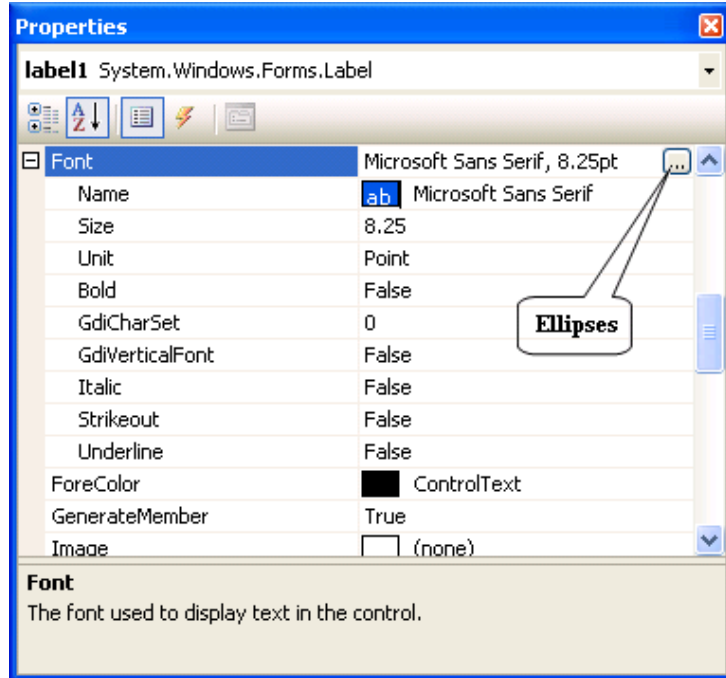
**Your Label control may now look something like the following:**



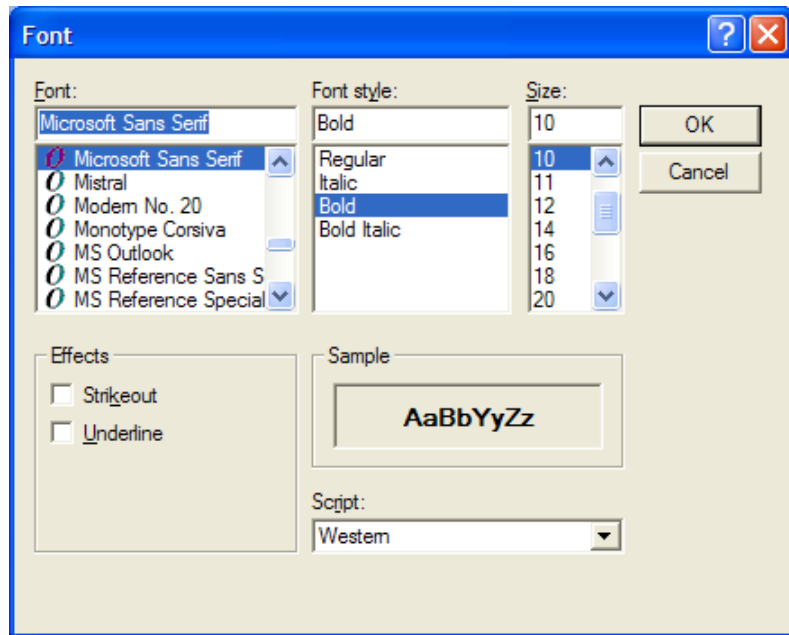
*and we want it to look like this:*



- 6. Let's change our **font** and **size** of our **Label** control.
  - a. Click the + sign to the left of the **Font** property in the Properties Window to display the **Font** properties, as shown below.



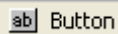
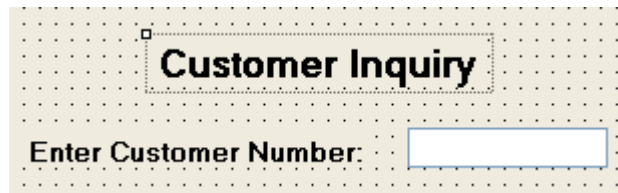
- b. You can change each individual property of the Font separately or you can click on the ellipses button to bring up the Font dialog box as shown below.



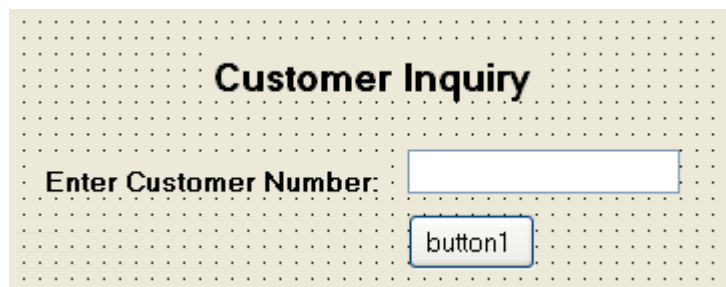
- c. For this example, we have chosen not to change the default Font. However, we will change the **Font style** to **Bold** and the **Size** to **10**.

 A Label

7. While we are using the **Label** control, let's go ahead and give this form a **Title** by adding another **Label** control.
  - a. Click and drag the **Label** control in the Toolbox onto the top area of the form.
  - b. Change its **Text** property to **Customer Inquiry**.
  - c. Click the + sign to the left of the **Font** property in the Properties Window, click on the ellipses and change the **Size** property to 14 and the **Font Style** to **Bold**.
  - d. Position the label to the desired location at the **Top** center of the form, similar to the following.

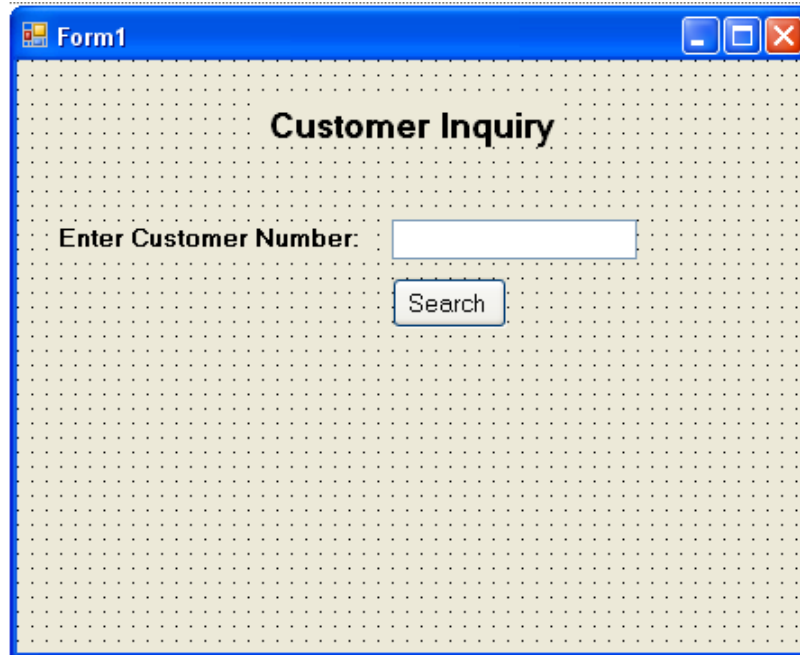
 Button

8. Next, we need to add a **Button** the user will click to initiate the customer search.
  - a. Click and drag the **Button** control onto the form underneath the **Customer Number** TextBox (or anywhere you desire).



- b. Change its **Name** property from **button1** to **btnSearch**.
- c. Change the **Text** property from **button1** to **Search**.
- d. Change the **Font Bold** property to **False** (if desired).
- e. Change the **Size** property to **62, 28**.

**At this point your form should now look similar to the following:**



9. We will be adding more elements to the form, but before we go any further, let's save the Windows form by selecting **File – Save Form1.vr**, or select **Ctrl+S**.

*You can tell whether a form needs to be saved by looking at the Form name in a Tab above the form and seeing if there is an asterisk (\*) after the form name, as shown below:*

**Form1.vr [Design]\***

The asterisk indicates that the form needs to be saved, or that a change(s) have been made to the form since it has been last saved.

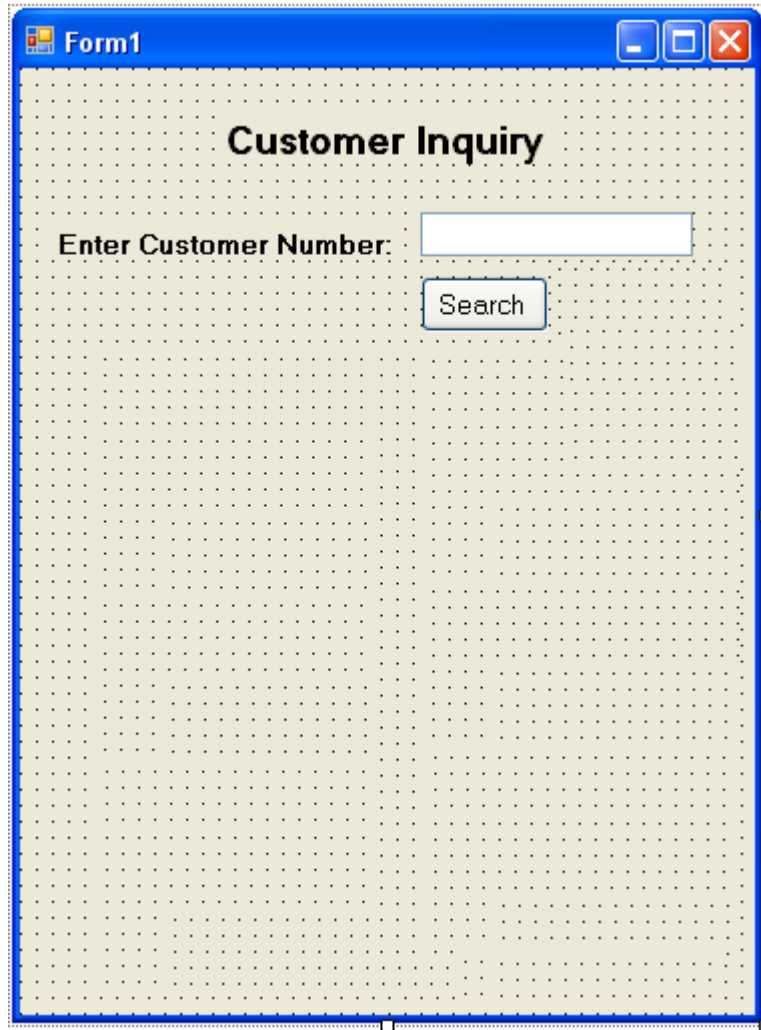
When you Save the form, notice that the “asterisk” disappears as shown below.

**Form1.vr [Design]**

*You can also select **Save All** (Shift+Ctrl+S), which will save ALL components of your Project, including the Solution, Project and Code.*

### Let's Continue Building our Form

At this point, we need to increase the size of the form so we will be able to see the customer's information. To do so, grab the square "handle" on the bottom center of the form with the mouse pointer and drag the form down. Your form should look like the following.

The image shows a screenshot of a Windows application window titled "Form1". The window has a blue title bar with standard minimize, maximize, and close buttons. The main area of the form has a light gray background with a fine grid pattern. At the top center of the grid, the text "Customer Inquiry" is displayed in a bold, black font. Below this, on the left side, is the text "Enter Customer Number:" followed by a white text box with a thin gray border. To the right of the text box is a rectangular button with the word "Search" written on it. The form is surrounded by a blue border, and small square handles are visible at the bottom and right edges, indicating it is in a design or development environment.

*Note:* You can also enter the size for your form. This example has set the **Size** property for the Form to **375, 507**.

Follow the steps below to create additional text boxes to display the customer's **Name**, **Address**, **City**, **State**, **Post Code**, **Phone**, and **Fax Number**, so the form looks like the following:

**A** Label

1. Add a Label for **Name**.
  - a. Click and drag the **Label** control in the Toolbox onto the form.
  - b. Change its **Text** property from **label3** to **Name:**.
  - c. Change the **Font - Bold** property to **False**.

*Name:*

**lab1** TextBox

2. Add a TextBox for **Name**.
  - a. Click and drag the **TextBox** control in the Toolbox onto the form.
  - b. Change its **Name** property from **textBox1** to **txtName**.
  - c. Change the **MultiLine** property to **True**.
  - d. Change the **Size** property to **184, 42**.
  - e. Change the **WordWrap** property to **True**.

*Name*

**A** Label

3. Add a Label for **Address**.
  - a. Click and drag the **Label** control in the Toolbox onto the form directly under the Name Label.
  - b. Change its **Text** property from **label4** to **Address:**.

*Address:*

**lab1** TextBox

4. Add a TextBox for **Address**.
  - a. Click and drag the **TextBox** control in the Toolbox onto the form directly under the Name TextBox.

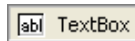
*Address*

- b. Change its **Name** property from **textBox1** to **txtAddress**.
- c. Change the **MultiLine** property to **True**.
- d. Change the **Size** property to **184, 42**.
- e. Change the **WordWrap** property to **True**.



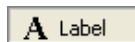
*City:*

5. Add a Label for **City**.
  - a. Click and drag the **Label** control in the Toolbox onto the form directly under Address Label.
  - b. Change its **Text** property from **label5** to **City:**.



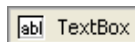
*City*

6. Add a TextBox for **City**.
  - a. Click and drag the **TextBox** control in the Toolbox onto the form directly under the Address TextBox.
  - b. Change its **Name** property from **textBox1** to **txtCity**.
  - c. Change the **Size** property to **136, 22**.



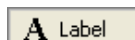
*State:*

7. Add a Label for **State**.
  - b. Click and drag the **Label** control in the Toolbox onto the form directly under City Label.
  - d. Change its **Text** property from **label6** to **State:**.



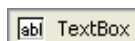
*State*

8. Add a TextBox for **State**.
  - a. Click and drag the **TextBox** control in the Toolbox onto the form directly under the City TextBox.
  - b. Change its **Name** property from **textBox1** to **txtState**.
  - c. Change the **Size** property to **136, 22**.



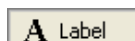
*Post Code:*

9. Add a Label for **Post Code**.
  - a. Click and drag the **Label** control in the Toolbox onto the form directly under State Label.
  - b. Change its **Text** property from **label7** to **Post Code:**.



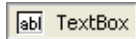
*Post Code*

10. Add a TextBox for **Post Code**.
  - a. Click and drag the **TextBox** control in the Toolbox onto the form directly under the State TextBox.
  - b. Change its **Name** property from **textBox1** to **txtPostCode**.



*Phone:*

11. Add a Label for **Phone**.
  - a. Click and drag the **Label** control in the Toolbox onto the form directly under Post Code Label.
  - b. Change its **Text** property from **label8** to **Phone:**.
  - c. Change the **Size** property to **136, 22**.



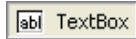
*Phone*

12. Add a TextBox for **Phone**.
  - a. Click and drag the **TextBox** control in the Toolbox onto the form directly under the Post Code TextBox.
  - b. Change its **Name** property from **textBox1** to **txtPhone**.
  - c. Change the **Size** property to **136, 22**.



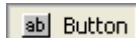
*Fax:*

13. Add a Label for **Fax**.
  - a. Click and drag the **Label** control in the Toolbox onto the form directly under Phone Label.
  - b. Change its **Text** property from **label9** to **Fax:**.



*Fax*

14. Add a TextBox for **Fax**.
  - a. Click and drag the **TextBox** control in the Toolbox onto the form directly under the Phone TextBox.
  - b. Change its **Name** property from **textBox1** to **txtFax**.
  - c. Change the **Size** property to **136, 22**.



15. Finally - add a Button to exit the application.
  - a. Click and drag a **Button** control in the Toolbox onto the form and place it next to the **Search** button.
  - b. Change its **Name** property from **button1** to **btnExit**.
  - c. Change its **Text** property to **Exit**.
  - d. Change the **Size** property to **64, 28** to match **Search** (if desired).

16. If your form looks like mine right now, we will need to fix the alignment of the Labels and TextBoxes so they appear in straight alignment on the left, and fix the vertical spacing, so there is the same amount of space between each field.
  - a. With your mouse, draw a “**Box**” around all of the Label controls, as shown below.

- b. Select **Format – Align - Lefts**. All labels will align to the left to match the alignment of the first field, **Name**.
  - c. Next, “draw” around all of the TextBox fields.
  - d. Select **Format – Align - Lefts**. All TextBoxes will align to the left to match the alignment of the first field, **txtName**.
  - e. Now, let’s change the vertical alignment, so each field has the same amount of space between the fields.
    - Draw a “box” around the Label controls and select **Format – Vertical Spacing – Make Equal**.
  - f. Draw a “box” around the TextBox controls and select **Format – Vertical Spacing – Make Equal**.
17. Your completed Windows Form should now look similar to the following:

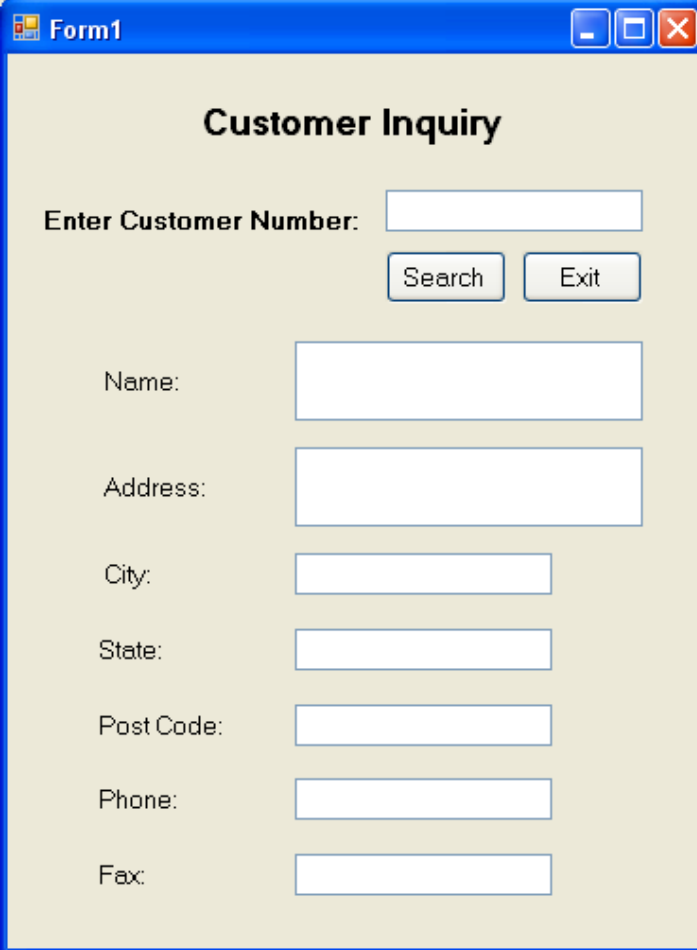
## Running the Windows Application

### Let's Run the Windows Application

1. Run your Windows application by selecting **Debug - Start**, or by simply pressing **F5**.

*Running the application will "save" the form with the same name.*





Your code will compile and your form will display on your screen as follows:





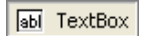
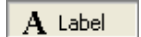

The screenshot shows a Windows application window titled "Form1". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main content area is light beige and contains the following elements:

- A title "Customer Inquiry" centered at the top.
- A label "Enter Customer Number:" followed by a text input field.
- Two buttons, "Search" and "Exit", positioned below the input field.
- A series of labels and text input fields stacked vertically: "Name:", "Address:", "City:", "State:", "Post Code:", "Phone:", and "Fax:".

*Because we have not added any additional code, this form does not do anything yet! We will add code in the next step.*

2. To stop your Windows application from running, simply click the  in upper right hand corner of the form. In Visual Studio, while the project is running, you will also see the pause  and stop  buttons in the Toolbar. Clicking on the stop  button will also stop the application from running.

## Step 2 Summary

To	Do This	Button/Keys
Create a Windows application	Select <b>File - New – Project - Visual RPG Projects - Windows Application.</b>	
View the Toolbox	Select <b>View – Toolbox.</b>	<b>Ctrl+Alt+X</b>
View the Properties Windows of a Control	Select the control, then do any of the following: <ul style="list-style-type: none"> <li>• Select <b>View – Properties Window.</b></li> <li>• Press F4.</li> <li>• Select the Properties Window icon in Toolbar.</li> </ul>	 
Insert a Text Box	Select the <b>TextBox</b> control and drag and drop onto the Windows form.	
Insert a Label	Select the <b>Label</b> control and drag and drop onto the Windows form.	
Insert a Button	Select the <b>Button</b> control and drag and drop onto the Windows form.	
Resize a Control	Click on the control, then click one of the square "handles" with the mouse pointer to resize to the desired dimensions.	
Align a group of controls	Select the appropriate option from the following aligning and sizing options: <p style="text-align: center;"> <b>Format – Align</b>  <b>Format – Horizontal Spacing</b>  <b>Format – Vertical Spacing</b> </p>	
Run a Windows Application	Select <b>Debug – Start.</b>	<b>F5</b>
Save a Windows Form	Select <b>File – Save.</b>	<b>Ctrl+S</b>
Save all components of a Project	Select <b>File – Save All.</b>	<b>Ctrl+Shift+S</b>

**This Page Intentionally Left Blank**

## Step 3: Responding to Events

---

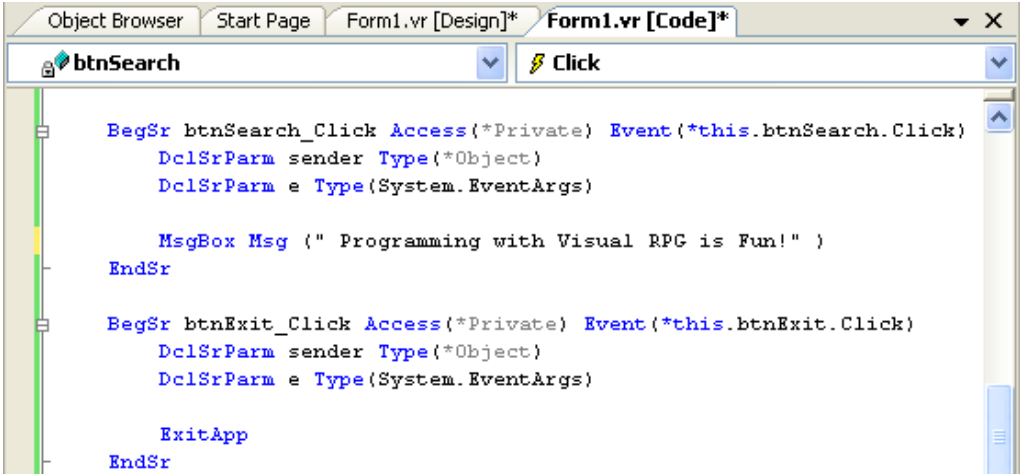
### What you will learn in Step 3:

- How to write code to respond to a button being clicked.

### Approximate Time to Complete Step 3:

Half an hour.

### What the code for Form1 will look like after completing Step 3:



```
Object Browser | Start Page | Form1.vr [Design]* | Form1.vr [Code]*
btnSearch | Click

BegSr btnSearch_Click Access(*Private) Event(*this.btnSearch.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)

  MsgBox Msg (" Programming with Visual RPG is Fun!" )
EndSr

BegSr btnExit_Click Access(*Private) Event(*this.btnExit.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)

  ExitApp
EndSr
```

## Adding Code to Form1.vr

At this point, we will write some code to show how this form will respond when the **Search** and **Exit** buttons are clicked.

### ✔ Let's Add Code to Form1.vr

1. Go to **Form1.vr[Design]** by clicking on the tab.

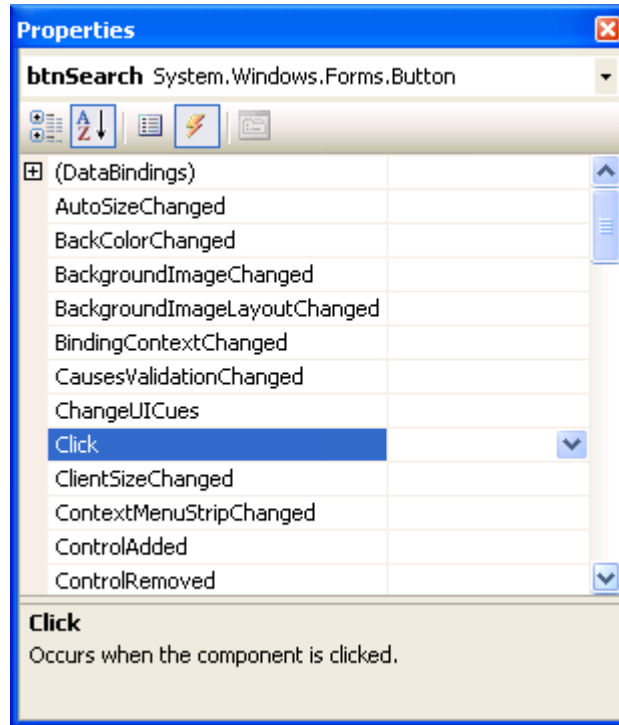
*What we want this form to do now is to respond to the clicking of the buttons. So first, let's write the code in the **Click** event of **btnSearch**.*

2. Click on the **Search** button (“**btnSearch**”) on **Form1.vr** to select it.
3. View your **Properties Window** (if it is not currently up, press **F4** to display it). You will see its properties displayed in the Properties Window, and it should look like the following:



4. However, we want to access the **Events** for this button, not its Properties, so click on the orange **lightning bolt** located at the top of your Properties Window.

This will display all the **events** (in alphabetical order) that can be set for that button. Near the top of the list is the "**Click**" event.



5. Double-click on the **Click** event name.

Notice that you automatically go the code page **Form1.vb [Code]**, and the code editor automatically generates the event subroutine **btnSearch\_Click**. The format is `<button name>_<event name>`. Your code generated for this event will look like this:

```

BegSr btnSearch_Click Access (*Private) Event (*this.btnSearch.Click)
  DclSrParm sender Type (*Object)
  DclSrParm e Type (System.EventArgs)
EndSr

```

Any code placed *within* this subroutine is executed whenever a user clicks the **Search** button while running the Windows application.

We need to add just **one line** of code that will display a windows message box on the screen. We will remove this line of code later.

6. Insert the following line before the **EndSr**:

```

MsgBox Msg ( "Programming with Visual RPG is Fun!" )

```

Your **btnSearch\_Click** routine should look like the following:

```

BegSr btnSearch_Click Access (*Private) Event (*this.btnSearch.Click)
  DclSrParm sender Type (*Object)
  DclSrParm e Type (System.EventArgs)

  MsgBox Msg ( "Programming with Visual RPG is Fun!" )
EndSr

```

7. Now we need to code an event for the **Exit** button. Follow the same steps as you did previously to add the **Click** event in the code editor for the **Exit** button.

Insert this line of code in the **Click** event subroutine for the **Exit** button.

`ExitApp`

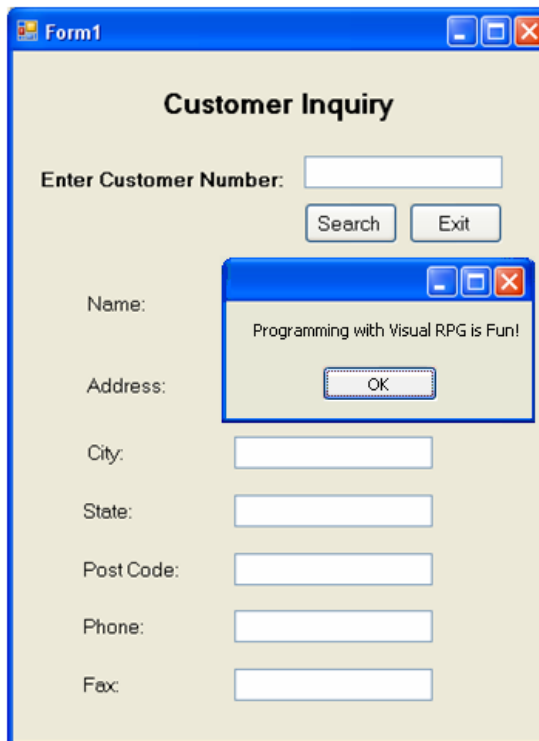
ExitApp is an operation code that will end the application.

Your **bntExit\_Click** routine should look like the following:

```
BegSr btnExit_Click Access(*Private) Event(*this.btnExit.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)

  ExitApp
EndSr
```


8. Let's Run the project and test our buttons. *You may want to 'Save' your code first by selecting Ctrl+S or Ctrl+Shift+S.*
- a. **Run** the project (F5). Click on the **Search** button. The Windows message box should appear, as shown below: Press **OK**.



- b. Now, click the **Exit** button. You Windows application will stop running, and return you to the code window.

***Great Job!*** Most of the work is now completed. All that remains is to take the customer number entered into the field and use it to **CHAIN** to the Customer Master file and display the data to the user.

## Step 3 Summary

To	Do This	Button/Keys
Access the events of a Toolbox control	Click on the <b>Lightning bolt</b> button in the <b>Properties</b> Window.	
Write code in the Click event for the <b>Search</b> button to display a message when clicked.	The following code caused the Windows message box to appear when the <b>Search</b> button was clicked.  <pre>MsgBox Msg( "Programming with AVR for Visual Studio 2008 is fun!" )</pre>	
Write code in the Click event for the <b>Exit</b> button to end the application when clicked.	The following code caused the application to end when the <b>Exit</b> button was clicked.  <pre>ExitApp</pre>	

**This Page Intentionally Left Blank**

## Step 4: Database Access

### What you will learn in Step 4:

- How to declare database and file objects.
- How to connect to a database and open a file.
- How to retrieve customer information and close a database and file.

### Approximate Time to Complete Step 4:

1 hour +.

### What the Code for Form1 will look like after completing Step 4:

At the end of Step 4, the application code for will look like the following:

```

BegSr Form1_Load Access(*Private) Event(*this.Load)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)
  Connect AppDB
  Open Cust
EndSr

BegSr btnSearch_Click Access(*Private) Event(*this.btnSearch.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)
  DclFld CustNo Type( *Packed ) Len( 9,0 )
  CustNo = txtCustNo.Text
  Chain Cust Key( CustNo ) Err( *Extended )
  If ( %Found )
    txtName.Text          = CMName
    txtAddress.Text       = CMAddr1
    txtCity.Text          = CMCity
    txtState.Text         = CMState
    txtPostCode.Text      = CMPostCode
    txtPhone.Text         = CMPhone
    txtFax.Text           = %EditW( CMFax, "0( )& - " )
  Else
    MsgBox Msg( "Customer Not Found" )
  EndIf
EndSr

BegSr btnExit_Click Access(*Private) Event(*this.btnExit.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)
  Close Cust
  Disconnect AppDB
  ExitApp
EndSr

```

## Adding Code to Form1 – Accessing a Database File

### Let's Add Code to Form1.vr

Now let's put the program to work by accessing the database for the customer data we're searching for.

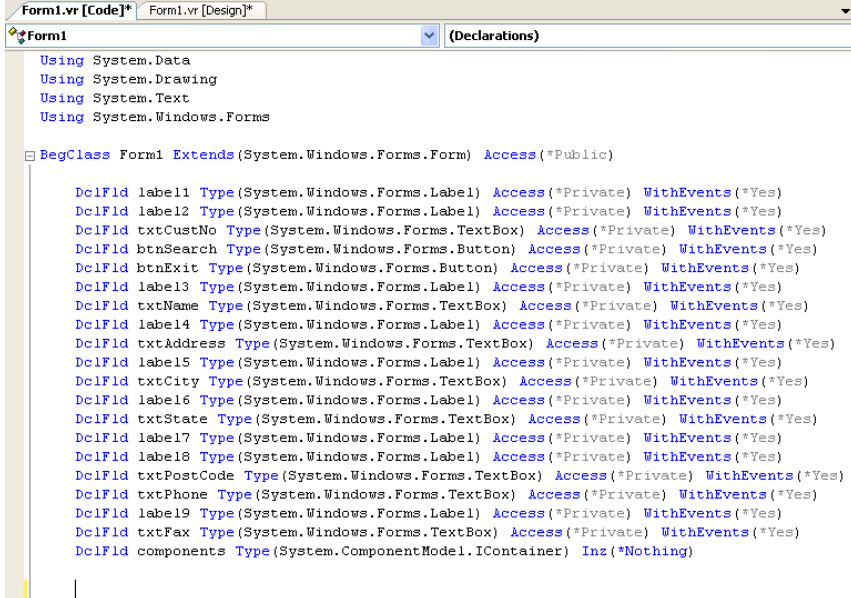
1. Go to Form1's code by double-clicking on the form or by using one of the following:

---

*Another way to get to the code for any form you are working on is to go to the **Solutions Explorer** window, right click the form, and then select **View Code**. Alternatively, you can select **View – Code** from the IDE menu, or press **F7**.*

---

2. Notice the code that is automatically generated by the Windows Designer (Using and DeclFid statements, etc). Do not worry about the Using statements for now, but notice the **DeclFid** statements. These were generated by the IDE as variables describing all the **objects** you placed onto **Form1**.



```

Form1.vr [Code]* Form1.vr [Design]*
Form1 (Declarations)
Using System.Data
Using System.Drawing
Using System.Text
Using System.Windows.Forms

Public Class Form1 Extends (System.Windows.Forms.Form) Access (*Public)

    DeclFid label1 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid label2 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid txtCustNo Type (System.Windows.Forms.TextBox) Access (*Private) WithEvents (*Yes)
    DeclFid btnSearch Type (System.Windows.Forms.Button) Access (*Private) WithEvents (*Yes)
    DeclFid btnExit Type (System.Windows.Forms.Button) Access (*Private) WithEvents (*Yes)
    DeclFid label3 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid txtName Type (System.Windows.Forms.TextBox) Access (*Private) WithEvents (*Yes)
    DeclFid label4 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid txtAddress Type (System.Windows.Forms.TextBox) Access (*Private) WithEvents (*Yes)
    DeclFid label5 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid txtCity Type (System.Windows.Forms.TextBox) Access (*Private) WithEvents (*Yes)
    DeclFid label6 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid txtState Type (System.Windows.Forms.TextBox) Access (*Private) WithEvents (*Yes)
    DeclFid label7 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid label8 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid txtPostCode Type (System.Windows.Forms.TextBox) Access (*Private) WithEvents (*Yes)
    DeclFid txtPhone Type (System.Windows.Forms.TextBox) Access (*Private) WithEvents (*Yes)
    DeclFid label9 Type (System.Windows.Forms.Label) Access (*Private) WithEvents (*Yes)
    DeclFid txtFax Type (System.Windows.Forms.TextBox) Access (*Private) WithEvents (*Yes)
    DeclFid components Type (System.ComponentModel.IContainer) Inz (*Nothing)
  
```

## Declaring Database and File Objects

- First, we will declare Database and File objects (i.e. **F-Specs**), and code them just **below** that large block of DclFids as noted in the diagram above.

Enter in the following lines:

```
// Database
DclDB Name( AppDB ) DBName( "DG Net Local" )

// Disk File
DclDiskFile Name      ( Cust )                +
                  Type  ( *Input )            +
                  File  ( "*Lib1/CMastNewL1" ) +
                  Org   ( *indexed )          +
                  ImpOpen ( *No )             +
                  DB    ( AppDB )
```

The first line of code (**DclDB**):

```
DclDB Name (AppDB) DBName ("DG Net Local")
```

declares a database object that will establish the connection to a Data Source.

- The **DBName** parameter describes the database name that delineates the database server, user ID, password, etc. In this case, whenever you reference the AppDB variable, you are actually referencing the connection to the “DG Net Local” database.
- The CMastNewL1 database is installed on your local machine and the [dictionary definition](#) can be found in Appendix A.

The **DclDiskFile** lines of code are all part of the same declaration, and they are responsible for creating an **F-Spec**.


- Notice the **Name** parameter is **Cust**. This is how you will refer to this file within your code.
- The **File** parameter refers to the **CMastNewL1** file in the library list (on the local database, in the Examples library).
- The **DB** parameter is the database name you specified previously. This informs the compiler that the format for this file can be found at the location specified within the AppDB database.
- The “+” is the AVR continuation character. All the DclDiskFile parameters can be on a single line if you prefer. Multiple lines are used here for readability.

This completes the variable declarations needed to accomplish our task, now let’s add some operational code.

- Now we need to add the **Form\_Load** event subroutine similar to the way we added the **Click** event for the **Search** and **Exit** buttons.

Select the **Form1** design view then click on **Form1** and, press the **F4** key to bring up the **Properties** window.

Another way to select **Form1** is to left-mouse click the drop-down arrow in the upper right hand corner of the **Properties** window and select **Form1** from the object list.

Click on the Events icon (lightning bolt ) to view the events for the form. Find the **Load** event in the list and double-click on it to add the load event to the code window. The code contained within this subroutine (shown below) executes **each** time the form is loaded by Windows.

```
BegSr Form1_Load Access (*Private) Event (*this.Load)
  DclSrParm sender Type (*Object)
  DclSrParm e Type (System.EventArgs)
EndSr
```

## Connecting to a Database

- Before you can do anything with the file, you must first **connect to the database**. Do not worry about the **DclSrParms** in the event. For now, just place all your code **below the DclSrParms and before the EndSr**. Enter the following command:

```
Connect AppDB
```

This one line establishes a connection with the database that you will be using throughout the application.

## Opening a File

- Next, you must **open** the file (Cust) that you will be reading from, so enter the following command:

```
Open Cust
```

The **Form1\_Load** event should now look like this.

```
BegSr Form1_Load Access (*Private) Event (*this.Load)
  DclSrParm sender Type (*Object)
  DclSrParm e Type (System.EventArgs)
  Connect AppDB
  Open Cust
EndSr
```

## Retrieving Customer Information (btnSearch\_Click)

- This next section of code will retrieve the customer information and display it on the windows form. Add the following code to the **btnSearch\_Click** event following the second **DclSrParm** statement. This code will execute when you click the **Search** button.

```

DclFld CustNo Type( *Packed ) Len( 9,0 )
CustNo = txtCustNo.Text
Chain Cust Key( CustNo ) Err( *Extended )
If ( %Found )
    txtName.Text      = CMName
    txtAddress.Text   = CMAAddr1
    txtCity.Text      = CMCity
    txtState.Text     = CMState
    txtPostCode.Text  = CMPostCode
    txtPhone.Text     = CMPhone
    txtFax.Text       = %EditW( CMFax, "0( )& - " )
Else
    MsgBox Msg( "Customer Not Found" )
EndIf

```

Before you begin, **remove the MsgBox Msg** added in Step 3.

First, we need to declare a **\*Packed** field that will be used to hold the customer number entered in the customer number text box. Then assign it the value from the customer number entered on the form. The remaining code contains the processing to access the database file for the number entered and **If** the record is found, display the customer information in the windows textboxes, **Else**, display an error message.

The **btnSearch\_Click** event subroutine should now look like the following:

```

BegSr btnSearch_Click Access(*Private) Event(*this.btnSearch.Click)
DclSrParm sender Type(*Object)
DclSrParm e Type(System.EventArgs)
DclFld CustNo Type( *Packed ) Len( 9,0 )
CustNo = txtCustNo.Text
Chain Cust Key( CustNo ) Err( *Extended )
If ( %Found )
    txtName.Text      = CMName
    txtAddress.Text   = CMAAddr1
    txtCity.Text      = CMCity
    txtState.Text     = CMState
    txtPostCode.Text  = CMPostCode
    txtPhone.Text     = CMPhone
    txtFax.Text       = %EditW( CMFax, "0( )& - " )
Else
    MsgBox Msg( "Customer Not Found" )
EndIf

```

## Closing a Database and File (btnExit\_Click)

Whenever the form is unloaded from memory, you must close the files and database connections. For this example, we will place the code necessary to do this in the **Click** event for the **Exit** button. (The **Form1\_Closed** event is also an optional event to place this code. This event fires when the form is unloaded from memory as well.)

8. In the **btnExit\_Click** event, place the following lines of code *before* the **ExitApp** operation code:

```

Close Cust
Disconnect AppDB

```

The `btnExit_Click` event should now look like this.

```
BegSr btnExit_Click Access(*Private) Event(*this.btnExit.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)
  Close Cust
  Disconnect AppDB
  ExitApp
EndSr
```

9. Now you can **Run** the entire application by pressing **F5**.

Your program will **connect** to the database and **open** your file when the application starts. Enter a customer number (see preceding note) in the customer number text box and click the **Search** button to **chain** to that record.

---

*In the Customer Master example file, all customer numbers end in '00'. Valid customer numbers are 100, 200, 300,...100000.*

---

Your finished results should look similar to the following (shown for customer number 300).

Form1

### Customer Inquiry

Enter Customer Number:

Name:

Address:

City:

State:

Post Code:

Phone:

Fax:

### **Congratulations!**

You have just completed your first Visual RPG for Visual Studio 2008 Windows Application!

## Step 4 Summary

To	Do This	Button/Keys
View Code	Select any of the following: <ul style="list-style-type: none"> <li>• Double-click on the form.</li> <li>• Go to the Solutions Explorer window, right-click on <b>Form1</b>, and then select <b>View Code</b>.</li> <li>• Select <b>View – Code</b> from the IDE menu.</li> <li>• Press <b>F7</b>.</li> </ul>	<b>F7</b>
Declare a Database	Use <b>DclDB</b> : <pre>DclDB Name( AppDB ) DBName( "DG Net Local")</pre>	
Declare a Disk File	Use <b>DclDiskFile</b> : <pre>DclDiskFile Name( Cust ) Type( *Input ) File( "**Libl/CMastNewL1" ) Org( *indexed ) ImpOpen( *No ) DB( AppDB )</pre>	
Connect to a Database	Use the <b>Connect</b> op code followed by the database (same as the Name parameter on the <b>DclDB</b> op code). <pre>Connect &lt;database name&gt;</pre>	
Open a Database file	Use <b>Open</b> : <pre>Open &lt;file&gt;</pre>	
Disconnecting from a Database	Use the <b>Disconnect</b> op code followed by the database (same as the Name parameter on the <b>DclDb</b> op code). <pre>Disconnect &lt;database name&gt;</pre>	
Close a Database file	Use <b>Close</b> : <pre>Close &lt;file&gt;</pre>	

## Additional Exercises

Now that you have successfully completed the tutorial, you might want to try these additional exercises on your own. Feel free to try other changes and experiment with the Form and control properties. For example, maybe add a logo, add a background color, etc.

- Change to Form1 title from **Form1** to **Customer Inquiry**.

---

*Hint:* Use Form1's **Text** property.

---

- Resize the TextBox controls to better conform to the length of the **CMastNewL1** database fields as shown in Appendix A.

---

*Hint:* Use the "handles" referenced in Step 2.

---

- Add code for the Phone TextBox to display the field contents similar to the Fax TextBox.

---

*Hint:* See **EditW** for possible editing options and the code added to set the value for `txtFax.Text`.

---

- Add the customer's second line of address (CMAAddr2) to the form, with or without another Label.

---

*Hint:* Add the **Label** (if used), add the **TextBox**, then add the code `txtAddress2.Text = CMAAddr2`.

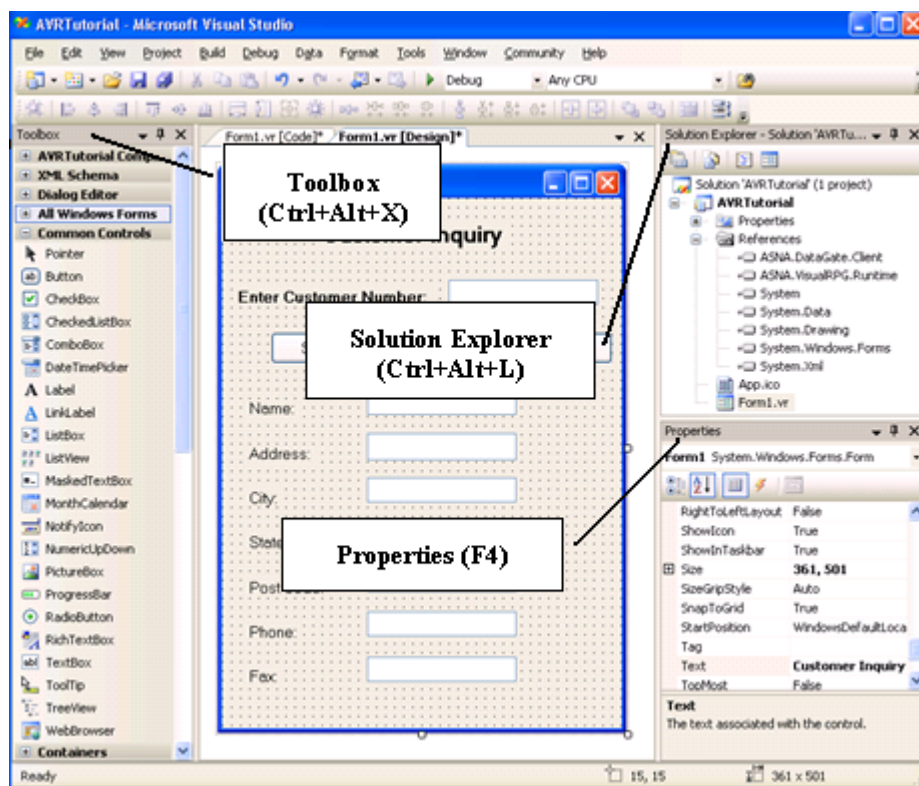
---

## Appendix A: Visual Studio 2008 101

There is a host of built-in tools in Visual Studio 2008 that aid in coding your project. This appendix will address the most commonly-used tools, giving brief descriptions of how to use them to your advantage.

### Forms Designer

The Forms Designer is a part of Visual Studio's IDE that allows you the programmer, to design your Windows application or Windows Form interface quickly and easily. Here is a typical view of the forms designer with descriptions of the three main windows and the shortcut keys to access them:



Refer to the **Visual Studio Quick Tour of the Integrated Development Environment** for more information. Also, refer to **How to: Arrange and Dock Windows** for details on how to dock the windows as shown above.

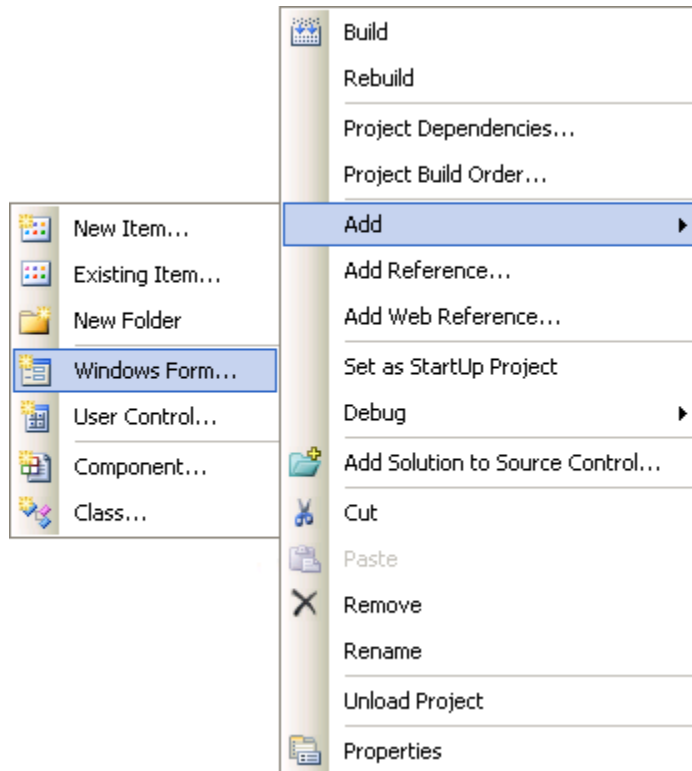
### Solution Explorer

The Solution Explorer is a navigational tool used in much the same way Windows Explorer is for Windows. From the Solution Explorer you can open and edit files within your project, and project's files within your solution. The files appear in the familiar tree-style layout. You can right-click on a file or folder in the Solution Explorer window for access to menu options that vary depending upon the object selected. For instance, menu options for **Form1.vr** in our AVR Tutorial contains the options shown below.

From this, you can open the file editing either the code or the design by selecting **View Code** or **View Designer**. You can also open the file in a different editor such as Notepad by selecting **Open With**. In addition, you can view the properties of the file by selecting the **Properties** option.

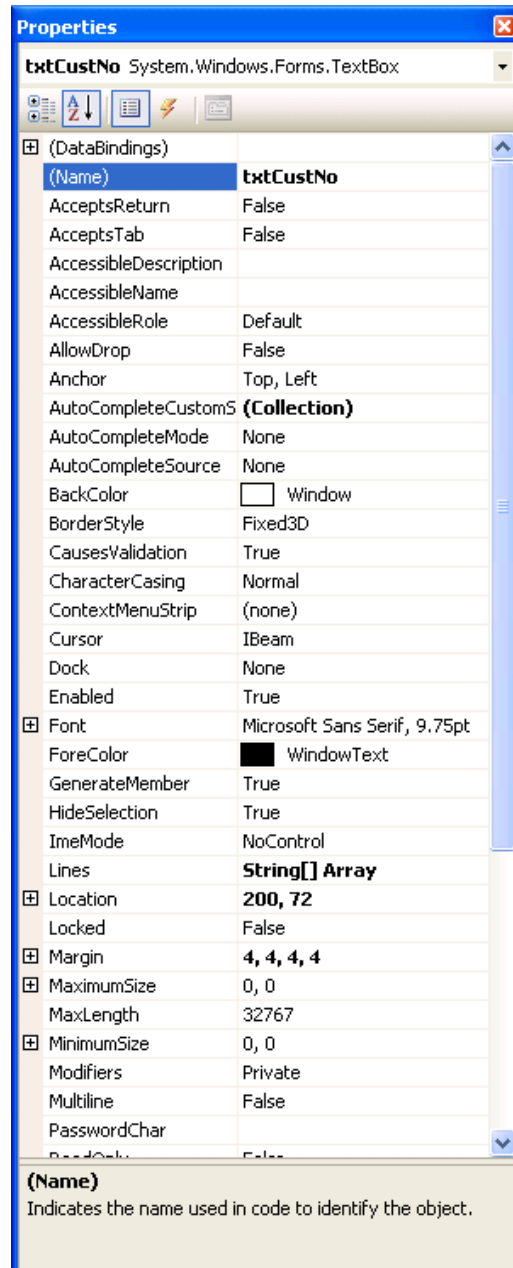


By right-clicking on the Project folder in Solution Explorer and selecting **Add**, you can add new or existing objects to your project as shown below:



## Properties

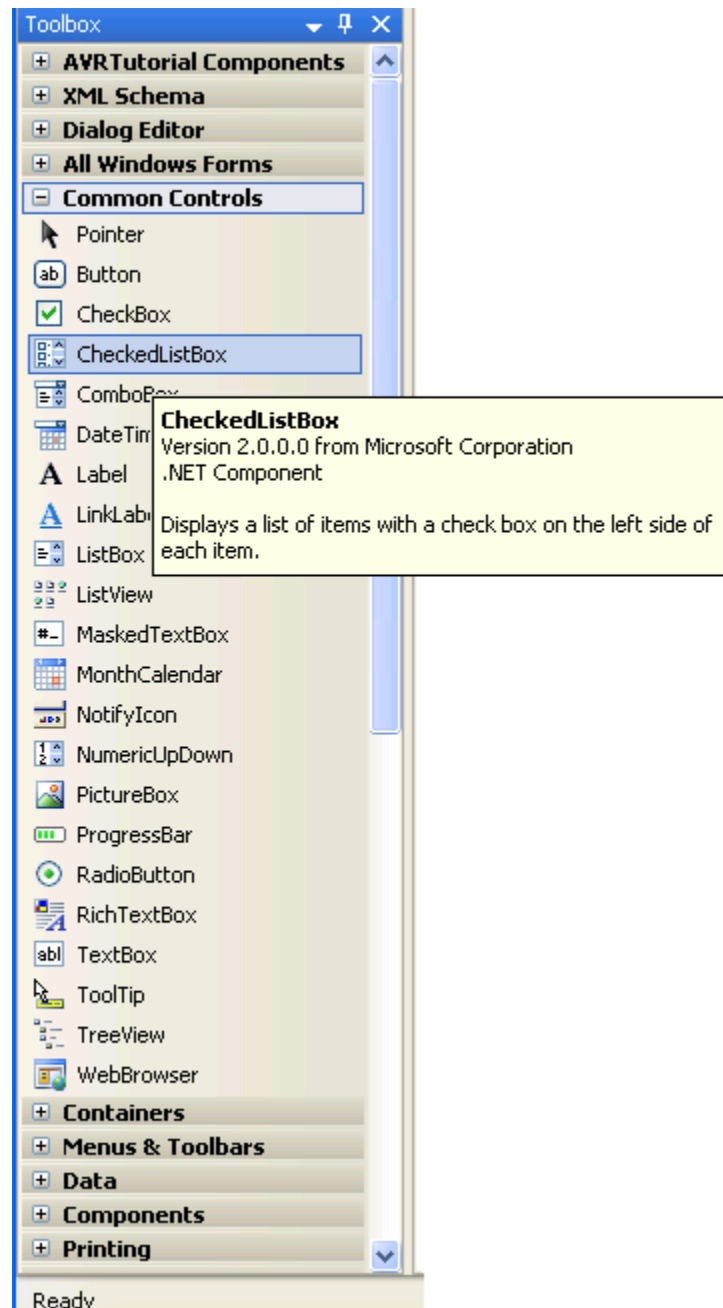
The properties pane provides a means to view and change the properties of controls used on your Windows forms. When you select a control in Design view, the Properties window automatically populates with all the Properties for that control. Here is a partial example of what it may look like for a TextBox control:



Here you can adjust virtually every configurable aspect of the control.

## Toolbox

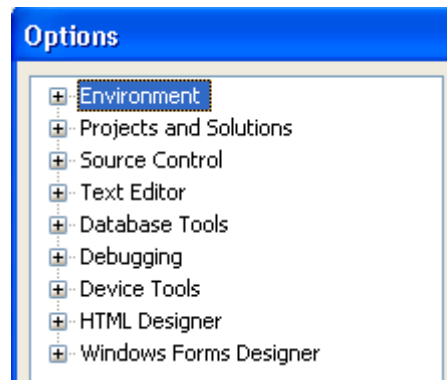
The Toolbox is a vital addition to the Forms designer because it holds all the common controls with which to populate your form. From the Toolbox, click a control component and drag it onto your form. The code behind is automatically generated. The Toolbox for **Common Controls** from our project looks like this:



Notice in the above example the component description displayed when you move your cursor over a component (**CheckListBox** above).

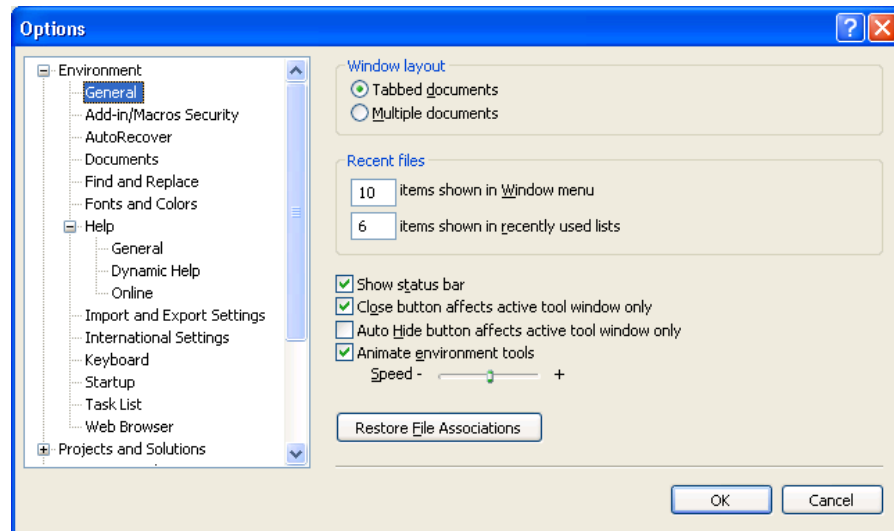
## Tools - Options

Several customized setting options are available to you through the IDE's Tools menu. If you click on **Tools - Options**, for the following settings:



Most of the changes you will probably want to make will be to the environment itself, customizing it to your liking. Within the environment folder you have many different options, but we will only cover the most commonly used ones here. For help on any of these options, use the help icon or press **F1**.

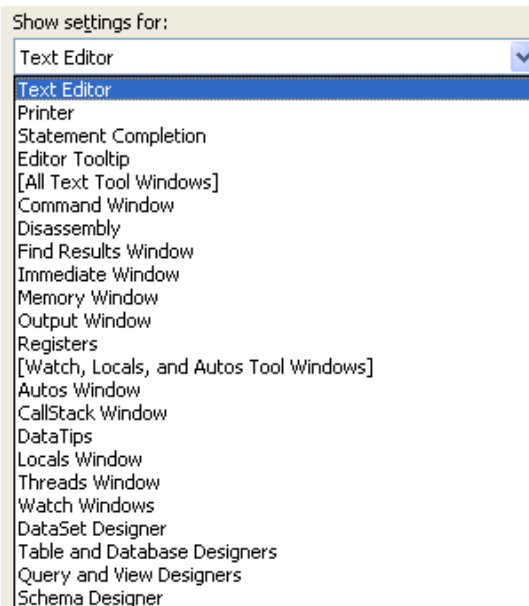
### Tools – Options – Environment - General



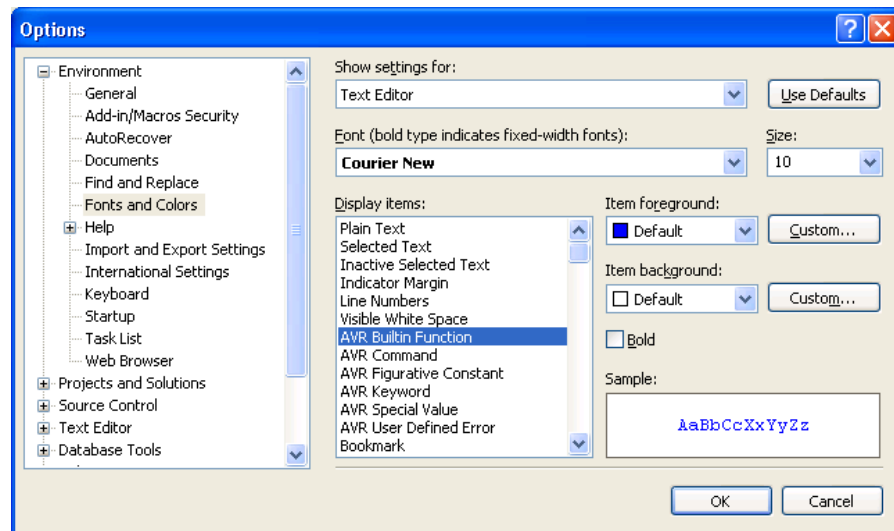
Most of the **General** options above are self-explanatory. An important one to mention here is the **Window layout** options: **Tabbed documents** or **Multiple documents**. This indicates how open files will appear as you work. If you choose **Tabbed** documents, the open files appear as tabs along the top of the IDE providing access to your files by clicking on the corresponding tab. The **Multiple documents** option causes each file opened to appear in its own window within the IDE.

## Environment - Fonts and Colors

This option is useful for customizing fonts and colors within the IDE for the list from the list available using the drop-down box shown below.



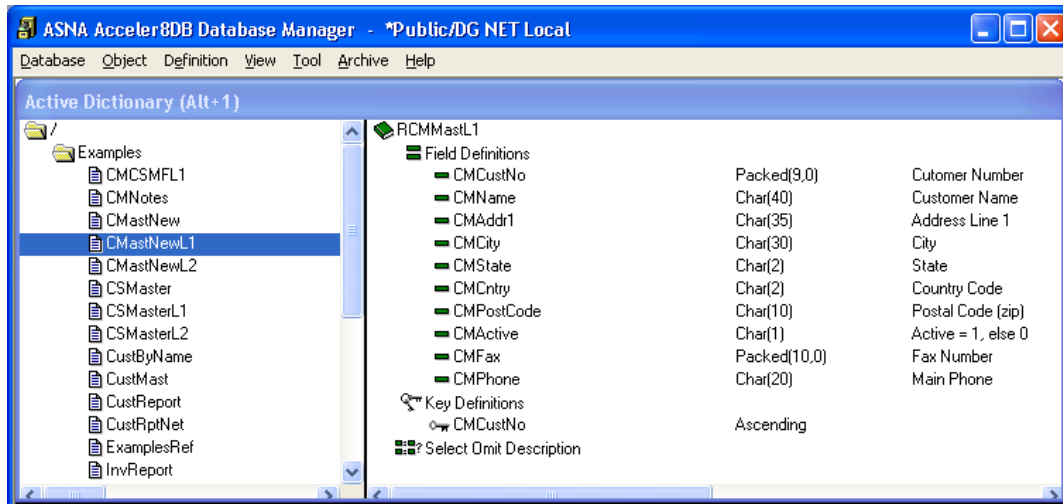
For our purposes, we will limit our discussion for those setting for the Text Editor shown below.



Here you can specify the font and color properties for the items shown in the **Display items** list. Each item can be set up with a specific font, font size, foreground and background colors, and bolded.

## CMastNewL1

The following is the file definition for the **CMastNewL1** file used in the tutorial.



**This Form Intentionally Left Blank**

## Index

Accessing			
events	28	aligning	22
help	12	button	17
Adding		label	15
button control	17	sizing	22
code to Form1	28	textbox	14
label control	15	Creating	
textbox control	14	new project	10
Additional exercises	40	Customer information	
Aligning		displaying	36
controls	22	retrieving	36
Appendix A		Database	
fonts and colors	46	connecting to	39
forms designer	41	declaring	39
general info	45	disconnecting from	39
properties	43	file definition	47
solution explorer	41	Database file	
toolbox	44	closing	39
tools - options	45	opening	39
Application		Database objects	
running	24	connecting to	36
ASNA Registration Assistant	7	declaring	35
AVR for Visual Studio 2008		disconnecting	37
installing for new users	6	DclDiskFile	
installing for previous users	6	coding	35
Button control		Declaring	
adding	17	database objects	35
Changing		file objects	35
font size	15	Design view	
Closing		forms	11
file	37	Disk file	
CMastNewL1		declaring	39
file definition	47	Displaying	
Code		customer information	36
adding to Form1	28	Docking windows	41
viewing	39	Events	
Code view		accessing	28
forms	11	F5	30
Coding		running application	24
btnSearch_Click event	29	File	
DclDiskFile	35	closing	37
Form_Load	35	file definition	
Connect		CMastNewL1	47
coding	36	File objects	
Connecting to		declaring	35
database	36	opening	36
Control		Font size	
resize	14	changing	15
Controls		Fonts and colors	
		about	46

appendix A	46	New project	
Form		creating	10
code view	11	New users	
Ctrl+S	18	installing AVR for Visual Studio 2008	6
design view	11	Opening	
resizing	19	file	36
saving	18	Options	
saving all	18	appendix A	45
setting grid	10	Previous users	
structure of	11	installing AVR for Visual Studio 2008	6
Form_Load		Project	
coding	35	saving all	18
Form1		Properties	
adding code to	28	about	43
Forms Designer		appendix A	43
about	41	viewing	14
appendix A	41	Resize	
General information		control	14
appendix A	45	Resizing	
Getting started	1	form	19
Grid		Retrieving	
setting for form	10	customer information	36
Help		Running	
accessing	12	the application	30
How to		Windows application	24
access the events of a control	31	Running application	
align a group of controls	25	pressing F5	24
close a database file	39	Saving	
connect to a database	39	form	18
Create a Windows application	25	Saving all	18
declare a database	39	Sizing	
declare a disk file	39	controls	22
disconnect from a database	39	Solution Explorer	
display a message	31	about	41
exit an application in code	31	appendix A	41
insert a button	25	Source files	2
insert a label	25	Step 1	
insert a text box	25	time to complete	5
open a database file	39	what you will learn	5
resize a control	25	Step 2	
run a Windows application	25	summary	25
save a windows application	25	time to complete	9
save all components	25	what it will look like	9
view code	39	what you will learn	9
view the properties of a control	25	Step 3	
view the toolbox	25	summary	31
Installing		time to complete	27
AVR for Visual Studio 2008 for new users	6	what it will look like	27
AVR for Visual Studio 2008 for previous users	6	what you will learn	27
Label control		Step 4	
adding	15	summary	39
Manual conventions	3	time to complete	33
		what it will look like	33

	<b>Index</b>	<b>51</b>
what you will learn	33	2
Structure		
windows forms	11	2
Summary		
step 2	25	1
step 3	31	3
step 4	39	2
Using the tutorial		2
Tabs		
All Windows Forms	13	14
view code	11	13
view form	11	
Textbox control		
adding	14	9
Time to complete		
step 1	5	27
step 2	9	33
step 3	27	
step 4	33	
Toolbox		
about	44	33
appendix A	44	
ToolBox		
about	12	41
All Windows formstab	13	45
viewing	13	14
Tutorial		
at the end of		10
at the end of each chapter		24
getting started		1
manual conventions		3
source files		2
using		2
Viewing		
properties		14
toolbox		13
What it will look like		
step 2		9
step 3		27
step 4		33
What you will learn		
step 1		5
step 2		9
step 3		27
step 4		33
Windows		
docking		41
layout options		45
properties		14
Windows application		
creating		10
running		24