VB-Motion
Getting Started Tutorials

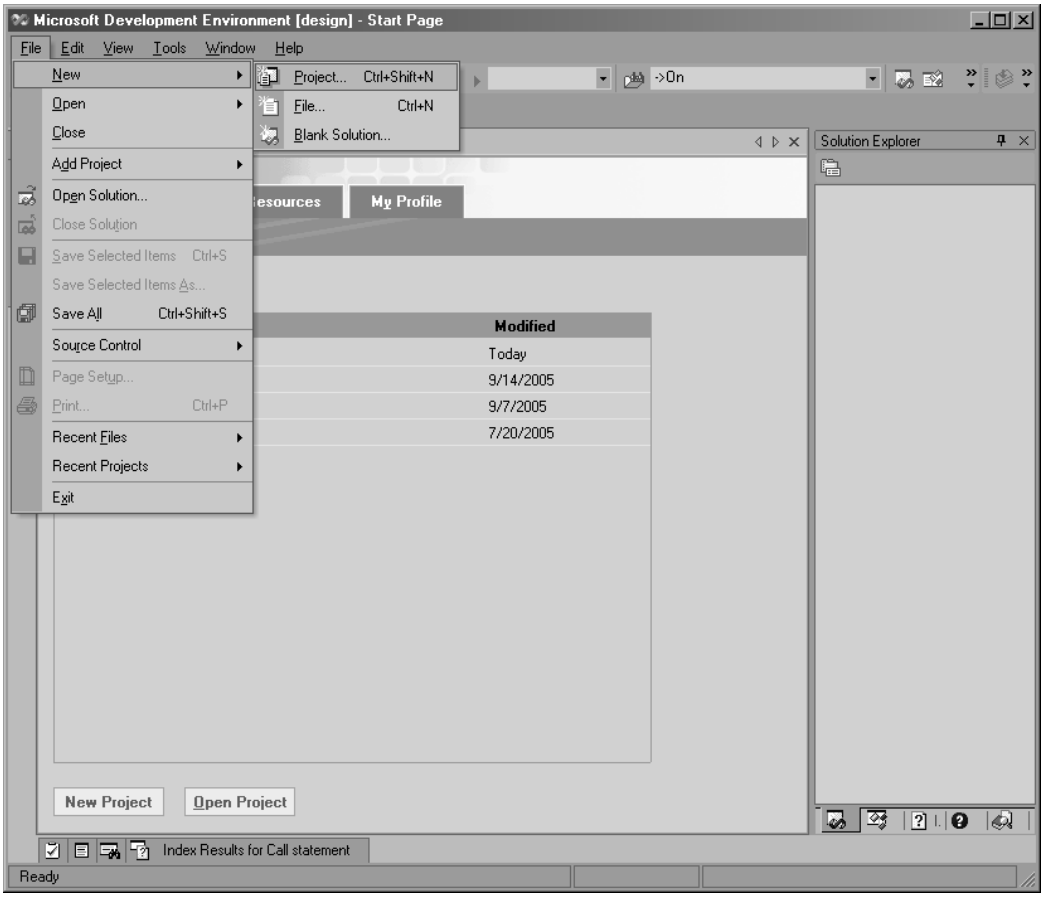
Performance Motion Devices, Inc.
55 Old Bedford Road
Lincoln, MA 01773

Revision 1.1, January 2007

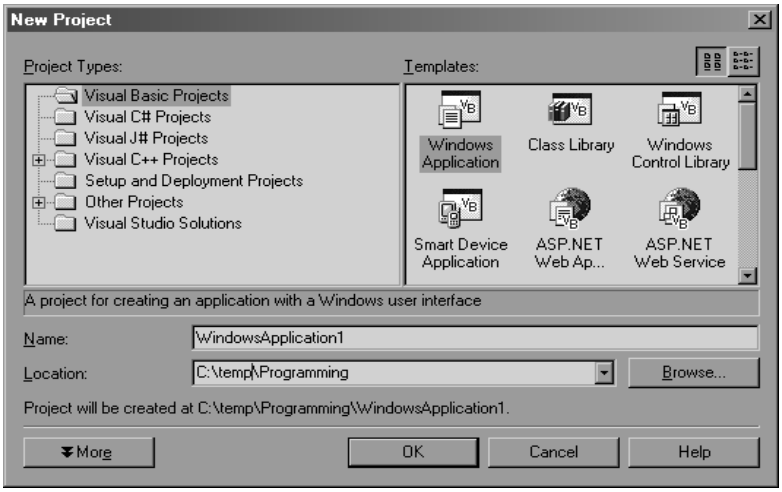
1. Creating a Visual Basic.NET Project

Open Visual Studio.NET.

Select **New Project** from the Visual Studio.NET **File** menu:

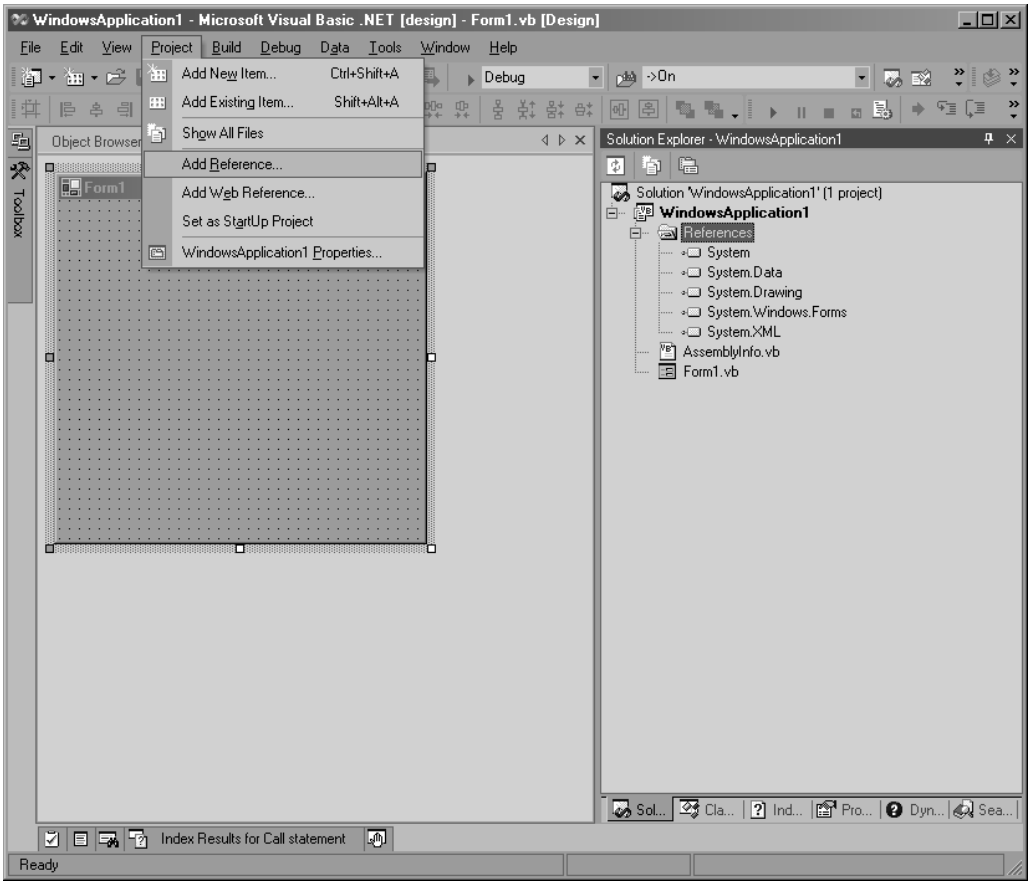


Select the **Visual Basic Projects** type from the **Project Types** tree and then **Windows Application** from the **Templates** list.



1.1 Adding Component References to the Project

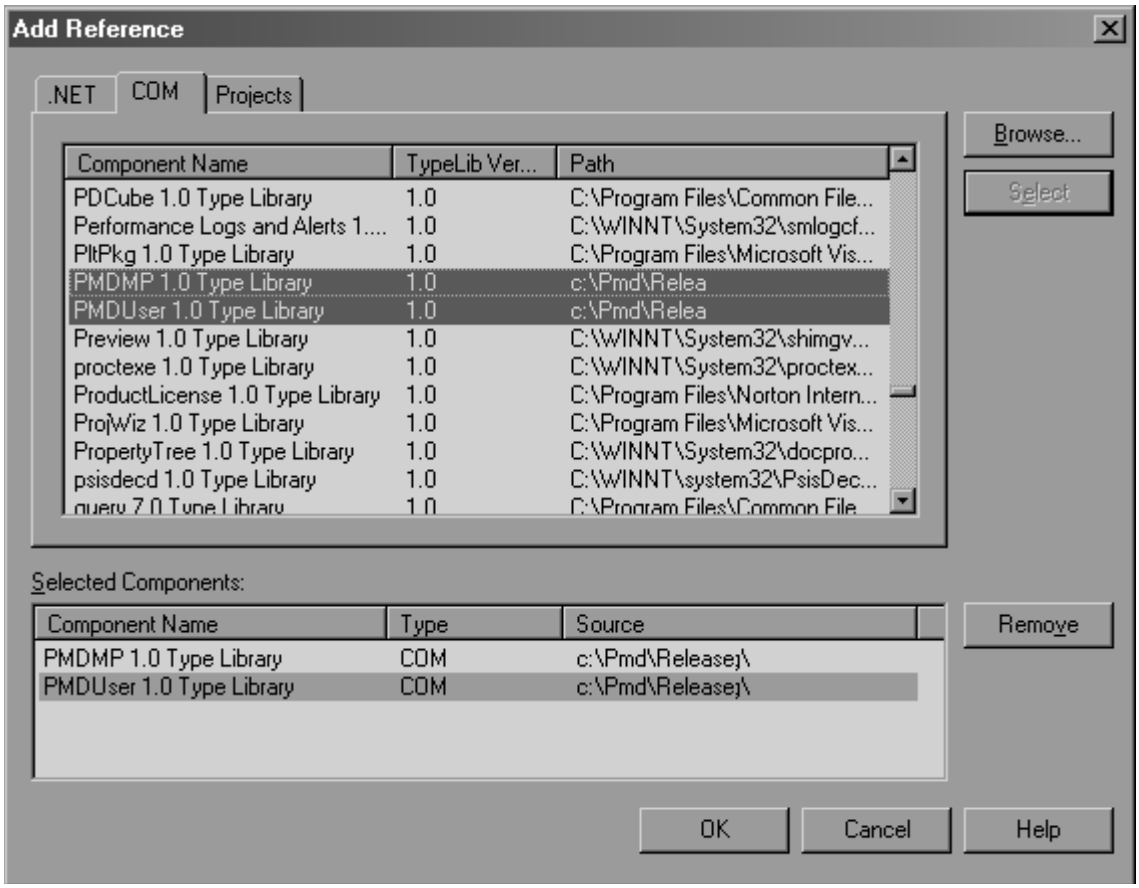
Select **Add Reference** from the **Project** menu item.



The **Add Reference** dialog box will be displayed.

Select the **COM** tab to display a list of available COM references.

Select **PMDMP 1.0 Type Library** and **PMDUSER 1.0 Type Library** from the list of **Available References**:

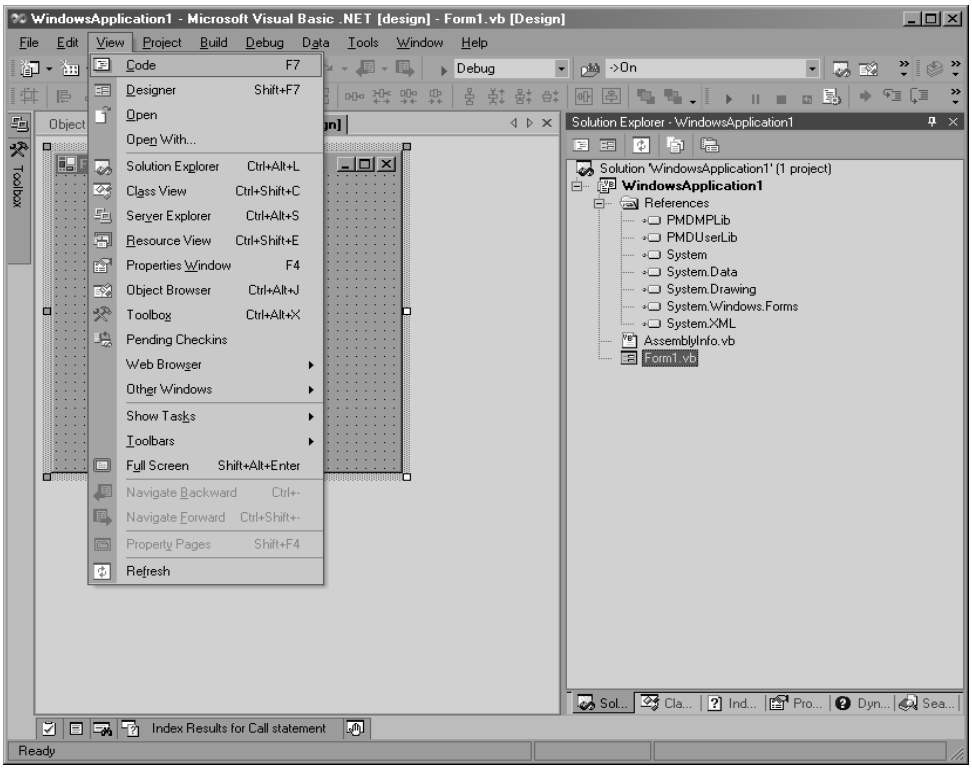


At this point, the programmer is ready to code within the default file (**Form1**) created by the Visual Basic 6 environment.

1.2 Viewing the Code Page

View the code page (from the Designer). If the code page is already in view, diregard this instruction and proceed to the next step.

Open the **Form1** code page by clicking on the **Code** menu item on the VB.NET **View** pulldown menu:



1.3 Adding Imports

Add the following statements to the **top** of the code page.

```
Imports PMDMPLib
Imports PMDUserLib
Imports System.Runtime.InteropServices
Imports System.Windows.Forms
```

This will allow the use of program control statements using these namespaces without the requirement of typing in fully qualified names.

1.4 Event Handling

Add a statement which will specify the event handling interface which will be implemented by this form.

```
Public Class FormMain
Inherits System.Windows.Forms.Form
Implements PMDMPLib.IPmdCommunicationEvent
```

NOTE: *This statement must be the first line of code after the Inherits… statement within the code page.*

The **IpmdCommunicationEvent** interface consists of three methods which allow the communication layer within the PMDMP and PMDUser components to communicate to connected clients.

IpmdCommunicationEvent consists of:

- OnChange() - The OnChange event is sent when a property of the object is modified.
- OnError(BSTR errmsg) - The OnError event is sent from the object when some type of error or invalid setup is detected. This error string can be useful in determining the problem.
- OnInterrupt(PMD_AXIS axis, short status) - OnInterrupt events are sent when the chip detects an interrupt condition on any axis and it is sent with the axis's interrupt status word.

Add the three IpmdCommunicationEvent interface method implementations. These are added at the program (Form1) scope.

```
Public Sub OnChange() _
Implements PMDMPLib.IPmdCommunicationEvent.OnChange
        'TODO: Add yourhandler code here
End Sub

Public Sub OnError(ByVal errmsg As String) _
Implements PMDMPLib.IPmdCommunicationEvent.OnError
        'TODO: Add yourhandler code here
End Sub

Public Sub OnInterrupt(ByVal axis As PMDMPLib.PMD_AXIS, ByVal status As Short) _
Implements PMDMPLib.IPmdCommunicationEvent.OnInterrupt
        'TODO: Add yourhandler code here
End Sub
```

1.5 Adding Component Object Variables

Add class variables to hold instances of the communication layer object, the Magellan object and the axis object respectively (a Serial communication driver is shown by default). These are added at the class (Form1) scope.

```
Private g_driverSerial As CommunicationSerial
Private g_object As MagellanObject
Private g_axis1 As MagellanAxis
```

1.6 Creating COM Objects

The following code is added within the **Form_Load()** subroutine.

Create an Serial communication driver object instance and set the COM port to COM1 (Note that other communication drivers may have differing methods to allow for specific driver setup).

Create a new Serial driver

```
g_driverSerial = New CommunicationSerial
```

Set the COM port to COM1

```
g_driverSerial.HostCOM = 1
```

Create a new MagellanObject instance and attach the Serial driver.

> **NOTE:** *The Call keyword has been eliminated from Visual Basic.NET.The use of parentheses around the argument list is required in all Function and Sub calls.*

Create a MagellanObject instance

```
g_object = New MagellanObject
```

Attach the Serial driver

```
g_object.SetupCommunication(g_driverSerial)
```

1.7 Registering the Event Handlers

Register the form as an event listener. This will hook the implemented IpmdCommunicationEvent interface to the object instance.

```
g_object.EventListenerRegister(Me)
```

1.8 Setting Up Tracing

Setup trace parameters.

Set the trace buffer wrap mode to a one time trace

```
g_object.TraceMode = PMD_TRACE_MODE_ONE_TIME
```

Set the processor variables that we want to capture

```
g_object.SetTraceVariable( _
PMD_VARIABLE_ID_1, PMD_AXIS_1, PMD_TRACE_VARIABLE_ACTUAL_POSITION)
g_object.SetTraceVariable( _
PMD_VARIABLE_ID_3, PMD_AXIS_1, PMD_TRACE_VARIABLE_ACTUAL_VELOCITY)
g_object.SetTraceVariable( _
PMD_VARIABLE_ID_4, PMD_AXIS_1, PMD_TRACE_VARIABLE_COMMANDED_VELOCITY)
```

Set the trace to begin when we issue the next update command

```
g_object.SetTraceStart( _
PMD_AXIS_1,PMD_TRACE_CONDITION_NEXT_UPDATE,_
PMD_EVENT_STATUS_MOTION_COMPLETE, True)
```

Set the trace to stop when the MotionComplete event occurs

```
g_object.SetTraceStop( _
PMD_AXIS_1, PMD_TRACE_CONDITION_NEXT_UPDATE,_
PMD_EVENT_STATUS_MOTION_COMPLETE, True)
```

1.9 Axis Control

Acquire an axis on the Magellan object (axis 1 illustrated)

```
g_axis1 = g_object.Axes(PMD_AXIS_1)
```

Set the profile parameters on the selected axis

```
g_axis1.ProfileMode = PMD_PROFILE_MODE_TRAPEZOIDAL
```

Set the axis position

```
g_axis1.Position = 200000
```

Set the axis velocity

```
g_axis1.Velocity = 20
```

Set the axis acceleration

```
g_axis1.Acceleration = 0.0625
```

Set the axis deceleration

```
g_axis1.Deceleration = g_axis1.Acceleration
```

Start the motion

```
g_axis1.Update
```

1.10 Unloading the COM Components

The programmer can unload the object, interface and axis components using the following statements, typically done at program end – in this example's case, during the form's **Closing** event handler.

```
Try
  If Not g_object Is Nothing Then g_object.EventListenerUnregister(Me)
  ' Release all COM resources
  If (Not g_driverSerial Is Nothing) Then Marshal.ReleaseComObject(g_driverSerial)
  If (Not g_axis Is Nothing) Then Marshal.ReleaseComObject(g_axis)
  If (Not g_object Is Nothing) Then Marshal.ReleaseComObject(g_object)
Catch ex As Exception
  MessageBox.Show("Unexpected error Closing main form! " + ex.Message)
End Try
```

In this sub, the **Try/Catch** block allows the form to unload gracefully in the event that Closing encounters an exceptional situation.

The call to EventListenerUnregister() deletes the client (this form) from the registered listeners to events sourced by the object and communication components.

The calls to set the object, driver and axis objects to Nothing effectively unload the components.

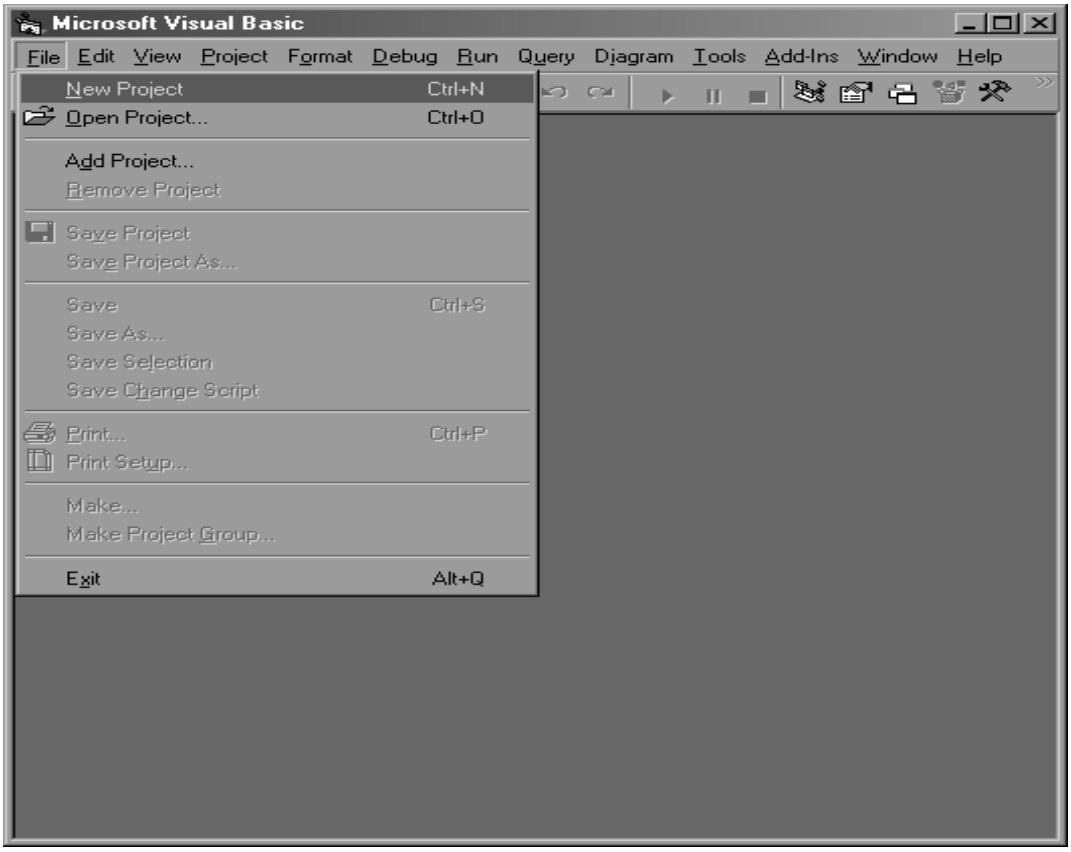
This page intentionally left blank.

2. Creating a Visual Basic 6 Project

This section describes creating an example project using VB 6.

Open Visual Basic 6.

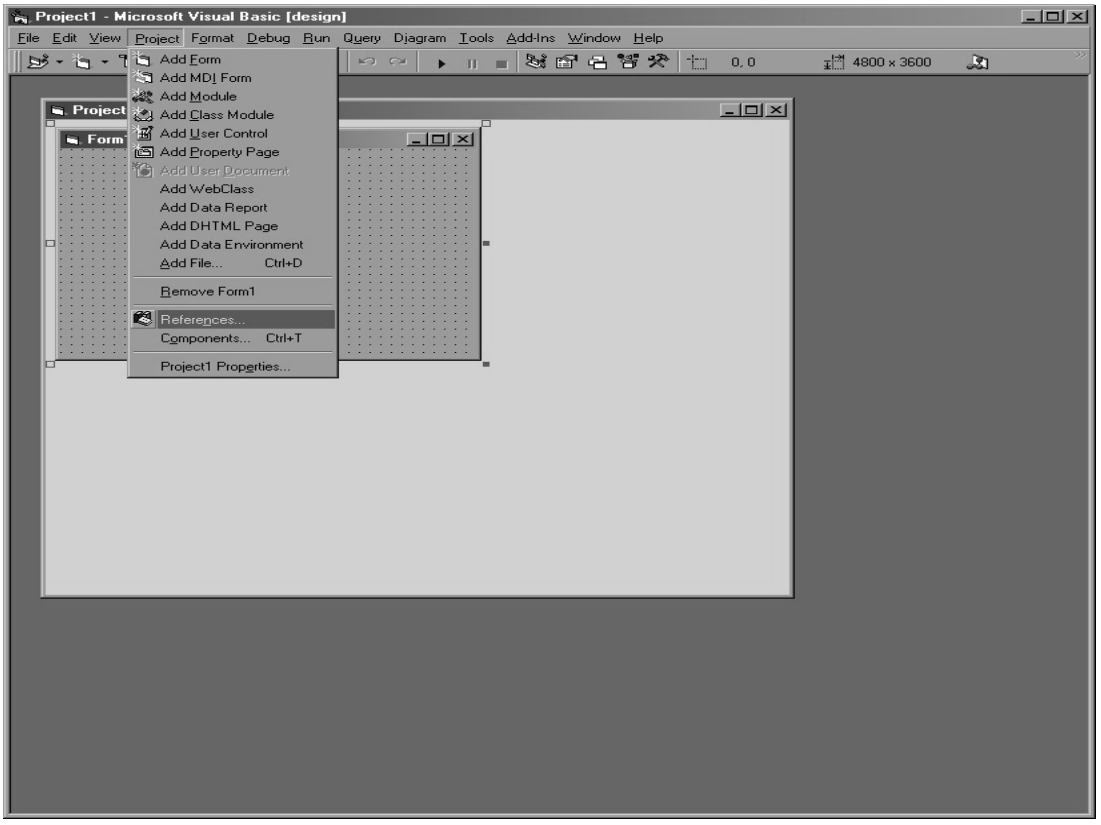
Select **New Project** from the Visual Basic 6 **File** menu:



Select **Standard Exe** from the **New Project** menu:

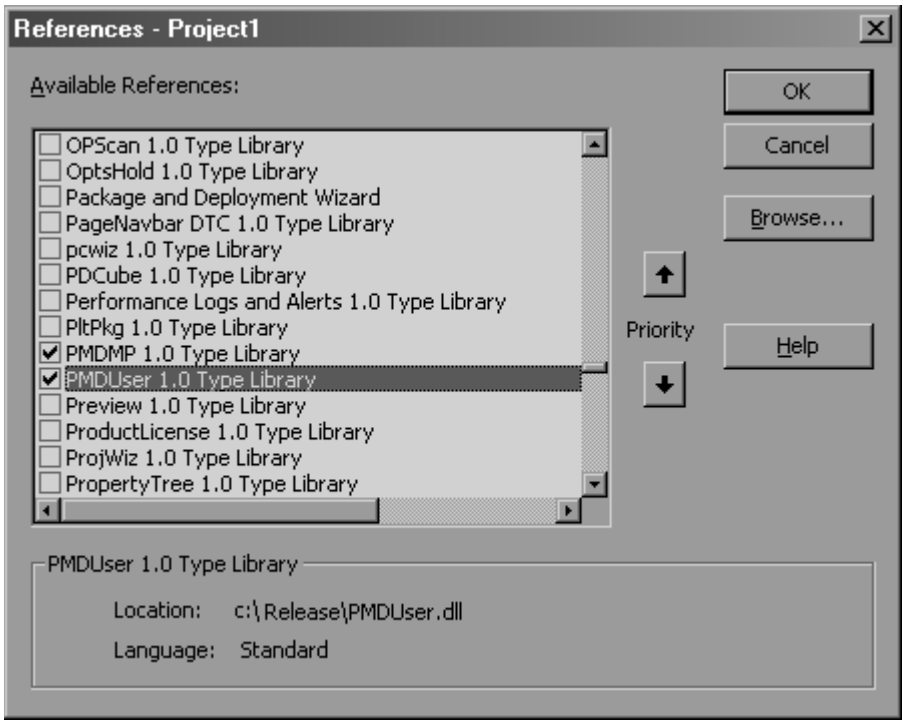
2.1 Adding Component References to the Project

Select **References** from the Visual Basic 6 **Project** menu:



The References dialog box will be displayed.

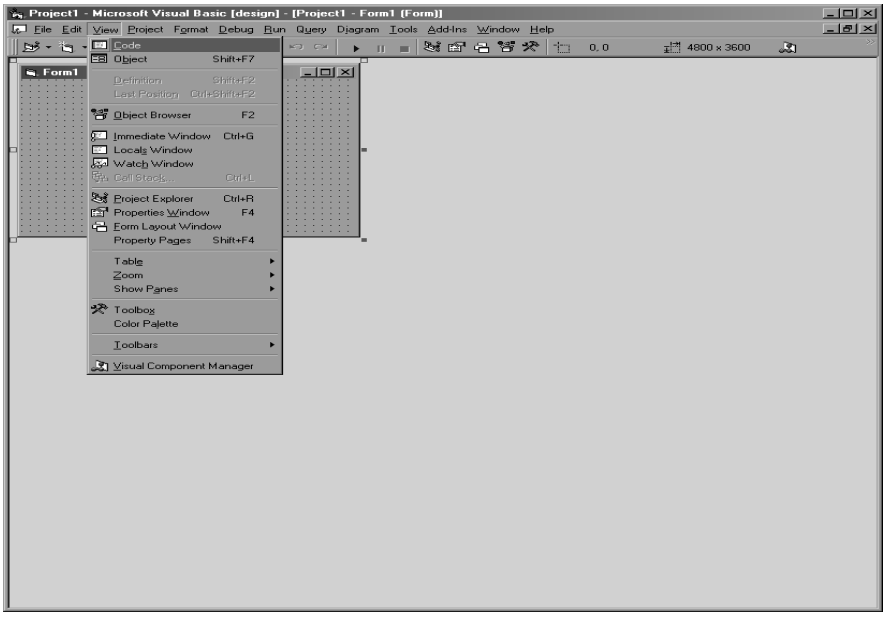
Select **PMDMP 1.0 Type Library** and **PMDUSER 1.0 Type Library** from the list of **Available References**:



At this point, the programmer is ready to code within the default file (**Form1**) created by the Visual Basic 6 environment.

2.2 Viewing the Code Page

View the code page (from the Designer). If the code page is already in view, disregard this instruction and proceed to the next step. Open the **Form1** code page by clicking on the **Code** menu item on the VB6 **View** pull- down menu:



2.3 Event Handling

Add a statement which will specify the event handling interface which will be implemented by this form.

```
Implements IpmdCommunicationEvent
```

NOTE: *This must be the first line of code within the code page.*

The **IpmdCommunicationEvent** interface consists of three methods which allow the communication layer within the PMDMP and PMDUser components communicate to connected clients.

IpmdCommunicationEvent consists three events which are described elsewhere:

OnChange().

OnError(BSTR errmsg).

OnInterrupt(PMD_AXIS axis, short status).

Add the three IpmdCommunicationEvent interface method implementations. These are added at the program (Form1) scope.

```
Private Sub IPmdCommunicationEvent_OnChange()
   'TODO: Add yourhandler code here
End Sub

Private Sub IPmdCommunicationEvent_OnError(ByVal ErrMsg As String)
   'TODO: Add yourhandler code here
End Sub

Private Sub IPmdCommunicationEvent_OnInterrupt( _
ByVal axis As PMDMPLib.PMD_AXIS, ByVal status As Integer) _
'TODO: Add yourhandler code here
End Sub
```

2.4 Adding Component Object Variables

Add class variables to hold instances of the communication layer object, the Magellan object and the axis object respectively (an Serial communication driver is shown by default). These are added at the program (Form1) scope.

```
Private g_driverSerial As PMDUserLib.CommunicationSerial
Private g_object As PMDMPLib.MagellanObject
Private g_axis1 As PMDMPLib.MagellanAxis
```

2.5 Creating COM Objects

The following code is added within the **Form_Load()** subroutine.

Create an Serial communication driver object instance and set the COM port to COM1 (Note that other communication drivers may have differing methods to allow for specific driver setup).

Create a new Serial driver

```
Set g_driverSerial = New PMDUserLib.CommunicationSerial
```

Set the COM port to COM1

```
g_driverSerial.HostCOM = 1
```

Create a new MagellanObject instance and attach the Serial driver.

> **NOTE:** *The use of Call is optional. If Call is used in the invocation of a subroutine then the use of parentheses around the argument list is required, if call is omitted then the use of parentheses around the argument list is forbidden. For example, the sub SetupCommunication can be called in one of two ways:*
>
> - *Call SetupCommunication(g_driverSerial)*
> - *SetupCommunication g_driverSerial*
>
> *All of the following code segments within this document will use the Call statement syntax.*

Create a MagellanObject instance

```
Set g_object = New PMDMPLib.MagellanObject
```

And attach the Serial driver to the object:

```
Call g_object.SetupCommunication(g_driverSerial)
```

2.6 Registering the Event Handlers

Register the form as a event listener. This will hook the implemented IpmdCommunicationEvent interface to the object instance.

```
Call g_object.EventListenerRegister(Me)
```

2.7 Setting Up Tracing

Setup trace parameters. Set the trace buffer wrap mode to a one time trace

```
g_object.TraceMode = PMD_TRACE_MODE_ONE_TIME
```

Set the processor variables that we want to capture

```
Call g_object.SetTraceVariable( _
PMD_VARIABLE_ID_1, PMD_AXIS_1, PMD_TRACE_VARIABLE_ACTUAL_POSITION)
Call g_object.SetTraceVariable( _
PMD_VARIABLE_ID_3, PMD_AXIS_1, PMD_TRACE_VARIABLE_ACTUAL_VELOCITY)
Call g_object.SetTraceVariable( _
PMD_VARIABLE_ID_4, PMD_AXIS_1, PMD_TRACE_VARIABLE_COMMANDED_VELOCITY)
```

Set the trace to begin when we issue the next update command

```
Call g_object.SetTraceStart( _
PMD_AXIS_1,PMD_TRACE_CONDITION_NEXT_UPDATE,_
PMD_EVENT_STATUS_MOTION_COMPLETE, True)
```

Set the trace to stop when the MotionComplete event occurs

```
Call g_object.SetTraceStop( _
PMD_AXIS_1, PMD_TRACE_CONDITION_NEXT_UPDATE,_
PMD_EVENT_STATUS_MOTION_COMPLETE, True)
```

2.8 Axis Control

Acquire an axis on the Magellan object (axis 1 illustrated)

```
Set g_axis1 = g_object.Axes(PMD_AXIS_1)
```

Set the profile parameters on the selected axis

```
g_axis1.ProfileMode = PMD_PROFILE_MODE_TRAPEZOIDAL
```

Set the axis position

```
g_axis1.Position = 200000
```

Set the axis velocity

```
g_axis1.Velocity = 20
```

Set the axis acceleration

```
g_axis1.Acceleration = 1 / 16
```

Set the axis deceleration

```
g_axis1.Deceleration = g_axis1.Acceleration
```

Start the motion

```
g_axis1.Update
```

2.9 Unloading the COM Components

The programmer can unload the object, interface and axis components using the following statements, typically done at program end – in this example's case, during the form's Unload subroutine.

```
Private Sub Form_Unload(Cancel As Integer)
   On Error Resume Next
   Call g_object.EventListenerUnregister(Me)
   Set g_driverSerial = Nothing
   Set g_axis1 = Nothing
   Set g_object = Nothing
End Sub
```

In this sub, the call **On Error Resume Next** allows the form to unload gracefully in the event that the Form_Unload encounters an exceptional situation.

The call to EventListenerUnregister() deletes the client (this form) from the registered listeners to COM events sourced by the Magellan object and communication components.

The calls to set the object, driver and axis objects to Nothing effectively unload the components.

For additional information, or for technical assistance, please contact PMD at (781) 674-9860.

You may also e-mail your request to support@pmdcorp.com

Visit our website at http://www.pmdcorp.com

Performance Motion Devices
55 Old Bedford Rd.
Lincoln, MA 01773