Application Note

Visual Basic DLL Interface to ASIC-200

The Visual Basic Interface to ASIC-200 is a Dynamic Link Library (DLL) called ASICVB.DLL. This interface allows you to pass data in real-time bi-directionally between ASIC-200 and Visual Basic 4.0/5.0/6.0. Microsoft's Visual Basic, supplemented with a library of OLE Control Extension (OCX) or ActiveX based controls, makes an excellent platform for designing Human/Machine Interfaces. The ASICVB.dll is an efficient interface for connecting your Visual Basic HMI to the ASIC-200 runtime controller. With the ASICVB.dll you can pass the following variables types to/from Visual Basic from/to ASIC-200: Real(double precision), String, Integer, Boolean, Byte, Word and Double Word using four simple function calls. The ASICVB.dll allows you to exchange user-defined types and array elements as well as individual variables. The ASICVB.dll works in conjunction with ASIC's Symsvr.dll to accomplish the interface.

You can supplement your Visual Basic package with any third party OCX or ActiveX controls, like MicroHelp's OLETools, in order to get better controls, such as knobs, dials, and gauges, than the ones you get with standard Microsoft Visual Basic.

Instructions :

1. Install ASIC in the C:\ASIC directory (default) or correct the path in the code in Step 4 below.
2. Install Visual Basic 4.0/5.0/6.0
3. Create an ASIC-200 application program. Make any variables that you want to send to or receive from Visual Basic global in the ASIC-200.
4. Create a Visual Basic Project and add the following Private Declarations in the General declaration area of the Code section of the Form. *Make sure they appear exactly the way they are shown below for it to work properly in Visual Basic*:

```
Private Declare Function Attach Lib "c:\asic\bin\asicvb.dll" Alias "#1" () As Long
Private Declare Function GetValue Lib "c:\asic\bin\asicvb.dll" Alias "#2" (ByVal name As String,
    var As Any) As Long
Private Declare Function SetValue Lib "c:\asic\bin\asicvb.dll" Alias "#3" (ByVal name As String,
    ByVal var As Variant) As Long
Private Declare Function GetArrayValue Lib "c:\asic\bin\asicvb.dll" Alias "#4" (ByVal name As
    String, ByVal subscript As Long, var As Any) As Long
Private Declare Function SetArrayValue Lib "c:\asic\bin\asicvb.dll" Alias "#5" (ByVal name as
    String, ByVal subscript As Long, ByVal var As Variant) As Long
Private Declare Function GetValueAsString Lib "c:\asic\bin\asicvb.dll" Alias "#6" (ByVal name As
    String, var As Any) As Long
Private Declare Function SetValueAsString Lib "c:\asic\bin\asicvb.dll" Alias "#7" (ByVal name As
    String,ByVal var As String) As Long
```

5. In your form's Initialize procedure add the following code:

```
status = Attach()
```

ASAP Inc. 100 North Main Street, Suite 235, Chagrin Falls, OH 44022
Telephone: (440) 247-9216 Fax: (440) 247-9218 Email: support@asapinc.com

If status = 0 Then
MsgBox ("Failed attach to ASIC-200.")
End If

6. Add the OCX controls to your Visual Basic form as needed.
7. In any VB procedure that requires the value of an ASIC variable, add code like the following example:

   retval = GetValue("ASICvarname", BasicVar)

   where: *retval* is defined as a boolean indicating success or failure of the GetValue( ).
   ASICvarname stands for the name of the ASIC variable to read. It must agree in case and spelling with ASIC.
   BasicVar stands for the name of the Basic variable to receive the information. The Basic variable should be of a type that is compatible with the ASIC variable.

   For arrays, use the GetArrayValue function as follows:

   retval = GetArrayValue("ASICarrayname",j,BasicVar)

   where: *retval* is defined as a boolean indicating success or faliure of the GetArrayValue( ).
   ASICarrayname is the name of the array declared in ASIC-200 application.
   j is the subscript (index) of the array element you wish to get value for.
   BasicVar stands for the name of the Basic variable to receive the information. The Basic variable should be of a type that is compatible with the ASIC variable.

8. In any VB procedure wishing to change an ASIC variable's value, add code like the following example:

   Dim bool As Long
   bool = True
   retval = SetValue("StartPB", bool)

   where: *StartPB* stands for the ASIC variable to change.
   bool stands for the Basic variable containing the value.

   For arrays, use the SetArrayValue function as follows:

   retval = SetArrayValue("ASICarrayname",j,BasicVar)

   Where: *ASICarrayname* stands for the array declared in an ASIC-200 application.
   j stands for the subscript (index) of the array element you wish to set value for.
   BasicVar is the variable declared with Visual Basic. It should have compatible type with ASIC-200.
9. *Private Declare Function GetValueAsString Lib "c:\asic\bin\asicvb.dll" Alias "#6" (ByVal name As String, var As Any) As Long*

This function takes a name of a symbol in ASIC-200 application as its first argument and returns the value of that symbol in ASIC-200 as a string regardless of the type of the symbol in ASIC-200. For example, a symbol of type INT in ASIC-200 with value of 1234 will be returned as "1234."

Private Declare Function SetValueAsString Lib "c:\asic\bin\asicvb.dll" Alias "#7" (ByVal name As String, ByVal var As String) As Long
This function takes name of a symbol in ASIC-200 applications as its first argument and the value of that symbol you wish to set in ASIC-200 as its second argument. Notice the second argument is a string value. For example, if you wish to set value of a REAL variable in ASIC-200 to 3.45, you will pass "3.45" as the second argument of the function.

The purpose of these functions is to allow type independence in getting and setting values from/to ASIC-200.

10. Using the values from ASIC-200 you can display the values or change the colors of your controls in order to accomplish your HMI functionality.
11. Start the ASIC-200 application running first and then start the Basic application, which will do the Attach(). You can have your Visual Basic application start ASIC-200 if you wish, but make sure that ASIC is running before you call Attach().
12. It is a good idea to Visual Basic timers and other techniques to minimize the number of requests for ASIC-200 to update variable values rather than issuing continuous requests in order to minimize your CPU utilization and improve performance.

Important notes about the ASICVB.dll

1. Basic variables used as booleans must be declared as Long. TRUE in Visual Basic is –1 and FALSE is 0. In ASIC-200, TRUE is 1 and FALSE is 0. Therefore, comparison of ASIC-200 boolean to Visual Basic boolean will not work.

2. Following table describes the data types in ASIC-200 and Visual Basic:

ASIC-200 Types	Visual Basic Types
INT/DINT	Long
WORD/DWORD	Long
REAL	Double
BYTE	Byte

STRING	String
BOOL	Long
TIME	Double*
DATE	String**
TOD	String**

*TIME type variables in ASIC-200 can be assigned a REAL value. So if you wish to a set timer value from Visual Basic, set a REAL variable in ASIC-200 with a corresponding Double in Visual Basic. Then, In ASIC-200, assign that REAL variable to the TIME variable. For example, a REAL in ASIC-200 with a value of 50.25 corresponds to 50 seconds and 250 milliseconds (T#50s250ms) when assigned to a TIME variable.

**DATE and TOD type variables in ASIC-200 can be assigned to a STRING type variable in ASIC-200 using just an "=" operator which can then be passed to Visual Basic. Thus, the corresponding variable in Visual Basic is declared as String. The reverse is not true. In other words, you cannot assign Date in Visual Basic to a string and set it in ASIC-200.

3. Constants cannot be used in the SetValue( ) procedure call.
4. Use GetValue( ) and SetValue( ) calls for used defined types. For example: *retval = GetValue ("ASICvar.member",BasicVar)*
5. Variables that are forced in ASIC-200 application through a watch window *cannot* be set by SetValue( ) or SetArrayValue( ) function calls for safety reasons.
6. Arrays of user defined types can be passed in Get/Set Value function or Get/Set ArrayValue function. Both of them work.