This file contains information about using the BIVBX library functions defined in the header file, bivbx.h, located in your \BC45\INCLUDE directory. If you are using VBX controls with your C, C++, or ObjectWindows applications, you will need to read this information so that you can use the correct VBX functions to initialize VBX support, return a VBX control handle, initialize a dialog window, handle events, and so forth.

For more information about using VBX controls in your C, C++, or ObjectWindows programs, see the online text file, VBX.TXT, which describes how to use VBXGEN, a utility program designed to generate a header file from a VBX control library.

TABLE OF CONTENTS

1. Initialization

BOOL VBXInit(HINSTANCE instance, LPCSTR classPrefix)

This function initializes VBX support for the program instance <instance> and must be called before any other VBX function. The <classPrefix> argument specifies the string prefix used when registering VBX window classes (NULL defaults to "BiVbx"). This function returns TRUE if successful or FALSE if unable to initialize.

void VBXTerm(void)

This function terminates VBX support for the current program instance. No other VBX functions should be called after this function.

BOOL VBXEnableDLL(HINSTANCE instApp, HINSTANCE instDLL)

This function enables VBX support for a DLL. It should be called prior to loading dialog resources from an instance other than the main program. It returns TRUE if successful or FALSE if an error occurs.

2.Controls

HCTL VBXGetHctl(HWND window)

This function returns the VBX control handle associated with the window <window> or NULL if <window> is not a valid VBX control.

HWND VBXGetHwnd(HCTL control)

This function returns the window handle associated with the VBX control <control> or NULL if <control> is not a valid VBX control.

HCTL VBXCreate(HWND windowParent, UINT id,
LPCSTR library, LPCSTR cls,
LPCSTR title, DWORD style,
int x, int y, int w, int h, int file)

This function creates a new instance of the control <cls> located in the VBX library <library>. The <style> argument specifies the control window style and can be set to 0 to use the default style. The <file> argument specifies a form file and is should be set to 0 for dynamically created controls. This function returns NULL if it is unable to load the VBX library and create the control. <x>, <y>, <w>, and <h> are related system coordinates.

3. Dialogs

BOOL VBXInitDialog(HWND window, HINSTANCE instance, LPSTR id)

This function is used to initialize a dialog window <window> loaded from a resource <id> (located in <instance>) by creating VBX controls for each child window of class VBControl located in the dialog template. It should be called by the dialog procedure when it receives the WM_INITDIALOG message. It returns TRUE if successful, or FALSE if an error occurs. Resource <id> must be of DLGINIT type.

4. Properties

int VBXGetNumProps(HCTL control)

This function returns the number of properties supported by the control <control> or -1 if an error occurs.

BOOL VBXGetProp(HCTL control, int index, LPVOID value)

This function retrieves the value of property <index> of control <control> and places it into the buffer located at <value>. It returns TRUE if successful, or FALSE if an error occurs.

ERR VBXGetPropByName(HCTL control, LPSTR name, LPVOID value)

This function retrieves the value of property <name> of control <control> and places it into the buffer located at <value>. It returns TRUE if successful, or FALSE if an error occurs.

For both VBXGetProp and VBXGetPropByName, <value> should be large enough to contain the property data type. Enumerated properties must have sizeof(value) >= sizeof(short).

int VBXGetPropIndex(HCTL control, LPCSTR name)

This function returns the index of the property <name> of control <control> or -1 if an error occurs.

LPCSTR VBXGetPropName(HCTL control, int index)

This function returns the name of property <index> of control <control> or NULL if an error occurs.

USHORT VBXGetPropType(HCTL control, int index)

This function returns the type (e.g. PTYPE_BOOL) of property <index> of control <control> or -1 if an error occurs. The property types are:

Type Name	C Type
PTYPE_CSTRING	HSZ
PTYPE_SHORT	short
PTYPE_LONG	long
PTYPE_BOOL	short
PTYPE_COLOR	COLORREF
PTYPE_ENUM	short
PTYPE_REAL	float
PTYPE_XPOS	long (twips)
PTYPE_XSIZE	long (twips)
PTYPE_YPOS	long (twips)
PTYPE_YSIZE	long (twips)
PTYPE_PICTURE	HPIC
PTYPE_BSTRING	HLSTR

BOOL VBXIsArrayProp(HCTL control, int index)

This function returns TRUE if the property <index> of control <control> is an array.

BOOL VBXSetProp(HCTL control, int index, LONG value)

This function sets the value of property <index> of control <control> to <value>. It returns TRUE if successful, or FALSE if an error occurs.

VBXSetPropByName(HCTL control, LPSTR name, LONG value);

This function sets the value of property <name> of control <control> to <value>. It returns TRUE if successful, or FALSE if an error occurs.

5.Events

When a VBX control generates an event, it sends a WM_VBXFIREEVENT message to its parent. The <lParam> argument of the message contains a far pointer to a VBXEVENT structure which describes the event:

```
typedef struct VBXEVENT
{
  HCTL        Control;
  HWND        Window;
  int         ID;
  int         EventIndex;
  LPCSTR      EventName;
  int         NumParams;
  LPVOID      ParamList;

} VBXEVENT, FAR * LPVBXEVENT, NEAR * NPVBXEVENT;
```

The <Control> member is the handle for the control that caused the event.
The <Window> member is the window handle for the above control.
The <ID> member is the control identifier for the above window.
The <EventIndex> member is the index into the event list for that control.

The <EventName> member is the name of the event (click, mouse move, etc.>.
The <NumParams> member is the number of arguments passed to the event.
The <ParamList> member is a pointer to an array (of size <NumParams>) of event arguments in reverse order (i.e. Arg0 == e->ParamList[e->NumParams-1]).

<type> VBX_EVENTARGNUM(event,type,index)

This macro retrieves an argument <index> of type <type> from VBX event <event>. Note that 0 is the first argument index. For example:

```
int x = VBX_EVENTARGNUM(event,int,0);
int y = VBX_EVENTARGNUM(event,int,1);
```

HLSTR VBX_EVENTARGSTR(event,index)

This macro retrieves a string argument <index> of type HLSTR from VBX event <event>. Note that 0 is the first argument index. For example:

```
HLSTR s = VBX_EVENTARGSTR(event,2);
```

int VBXGetEventIndex(HCTL control, LPCSTR name)

This function returns the index of event <name> of control <index> or -1 if an error occurs.

LPCSTR VBXGetEventName(HCTL control, int index)

This function returns the index of event <index> of control <control> or NULL if an error occurs.

int VBXGetNumEvents(HCTL control)

This function returns the number of events supported by the control <control> or -1 if an error occurs.

6. Methods

BOOL VBXMethod(HCTL control, int method, long far * args)

This function invokes method <method> on control <control> with arguments <args>. Note that this function is not normally called by application programs and is described here as a means of invoking custom control methods. It returns TRUE if successful or FALSE if an error occurs.

BOOL VBXMethodAddItem(HCTL control, int index, LPCSTR item)

This function invokes the standard "add item" method on control <control> where <index> is the index of the item to be added (<item>). The exact meaning of "add item" is dependent on the type of VBX control. It returns TRUE if successful or FALSE if an error occurs.

BOOL VBXMethodDrag(HCTL control, int action)

This function invokes the standard "drag" method on control <control> where <action> is one of the following:

0 - cancel a drag operation
1 - begin a drag operation
2 - "drop" the control at the current location

It returns TRUE if successful or FALSE if an error occurs.

BOOL VBXMethodMove(HCTL control, long x, long y, long w, long h)

This function invokes the standard "move" method on control <control>. The default behaviour for this method is to position the control at <x>, <y>, <w>, and <h>. It returns TRUE if successful or FALSE if an error occurs.

BOOL VBXMethodRefresh(HCTL control)

This function invokes the standard "refresh" method on control <control>. The default behaviour for this method is to update the contents of the control window before returning. It returns TRUE if successful or FALSE if an error occurs.

BOOL VBXMethodRemoveItem(HCTL control, int item)

This function invokes the standard "remove item" method on control <control> where <index> is the index of the item to be removed. The exact meaning of "remove item" is dependent on the type of VBX control. It returns TRUE if successful or FALSE if an error occurs.

7.Conversions

VBX controls make use of a combination of Twips and pixel measurements. The following functions are used to convert between these different measurement units:

SHORT VBXTwp2PixY(LONG twips)

This function converts a Y coordinate <twips> from twips to pixels.

SHORT VBXTwp2PixX(LONG twips)

This function converts an X coordinate <twips> from twips to pixels.

LONG VBXPix2TwpY(SHORT pixels)

This function converts a Y coordinate <pixels> from pixels to twips.

LONG VBXPix2TwpX(SHORT pixels)

This function converts an X coordinate <pixels> from pixels to twips.

8. Dynamic Strings

VBX controls make extensive use of moveable zero-terminated strings, or "dynamic strings". The following functions are used to manipulate those strings:

HSZ VBXCreateCString(HANDLE segment, LPSTR string)

This function creates a new string by allocating from the local heap in and initializing to <string>. It returns 0 if an error occurs.

LPSTR VBXGetCStringPtr(HSZ string)

This function returns a pointer to the contents of the dynamic string <string> or 0 if an error occurs.

HSZ VBXDestroyCString(HSZ string)

This function destroys the dynamic string <string>. It returns <string>.

LPSTR VBXLockCString(HSZ string)

This function locks the string <string> and returns a pointer to the contents or 0 if an error occurs.

void VBXUnlockCString(HSZ string)

This function unlocks the string <string>.

9. Pictures

VBX controls can support a variety of "picture" property types, including bitmaps, metafiles, and icons. These types are represented by a single structure which contains a union of the different types:

```
typedef struct PICTURE
{
 BYTE Type;
 union
 {
   struct
   {
     HBITMAP  Bitmap;
     HPALETTE Palette;
   } Bitmap;
   struct
   {
     HANDLE Metafile;
     int xExtent;
     int yExtent;
   } Metafile;
   struct
   {
     HICON Icon;
   } Icon;
 } Data;
 BYTE Unused0;
 BYTE Unused1;
 BYTE Unused2;
 BYTE Unused3;
} PICTURE, FAR * LPPICTURE, NEAR * NPPICTURE;

#define PICTURE_EMPTY     0
#define PICTURE_BMP       1
#define PICTURE_META      2
#define PICTURE_ICON      3
```

HPIC VBXCreatePicture(LPPICTURE picture)

This function creates and returns a new picture handle from a picture buffer <picture> or 0 if an error occurs. For example, the following code creates an icon picture:

```
PICTURE pic;
HPIC hpic;
pic.Type = PICTURE_ICON;
pic.Icon.Icon = LoadIcon( NULL, IDI_ASTERISK );
hpic = VBXCreatePicture( &pic );
```

void VBXDestroyPicture(HPIC pic)

This function decrements the reference count on the picture handle <pic> and destroys it if the count becomes 0.

HPIC VBXGetPicture(HPIC pic, LPPICTURE picture)

This function copies the contents of the picture handle <pic> into the picture buffer <picture> and returns <pic> if successful or 0 if an error occurs.

ERR VBXGetPictureFromClipboard(HPIC FAR *pic, HANDLE data, WORD format)

This function creates a new picture handle <*pic> from a clipboard data handle <data> and format <format> and returns non-zero if an error occurs. Valid clipboard formats include CF_BITMAP, CF_METAFILEPICT, CF_DIB and CF_PALETTE.

HPIC VBXReferencePicture(HPIC pic)

This function increments the reference count on the picture handle <pic> and returns <pic> if successful or 0 if an error occurs.

10. Basic Strings

VBX controls make use of moveable string buffers (not zero terminated), or "Basic strings." The following functions are used to manipulate those strings:

HLSTR VBXCreateBasicString(LPVOID buffer, USHORT len)

This function creates a Basic string of length <len> and initial contents of <buffer>. It returns 0 if an error occurs.

LPSTR VBXGetBasicStringPtr(HLSTR string)

This function returns a pointer to the contents of the Basic string <string> or 0 if an error occurs.

void VBXDestroyBasicString(HLSTR string)

This function destroys the Basic string <string>.

USHORT VBXGetBasicStringLength(HLSTR string)

This function returns the length of the Basic string <string> or 0 if an error occurs.

ERR VBXSetBasicString(HLSTR far * string, LPVOID buffer, USHORT len)

This function replaces the contents of the Basic string <string> with <len> bytes from <buffer>. It returns non-zero if an error occurs.

11. Arrays

The following functions illustrate how to get and set array properties:

```
typedef struct ELEMENTSTRUCT
{
 LONG   Value;                  /* Value to be transferred */
 USHORT NumElems;    /* Number of elements */
 struct
 {
  USHORT Type;                  /* Type of element (must be PTYPE_SHORT) */
  LONG   Index;                 /* Index of array element */
 } Element[1];

} ELEMENTSTRUCT, FAR * LPELEMENTSTRUCT, NEAR * NPELEMENTSTRUCT;

BOOL VBXGetArrayProp( HCTL control, int index,
                LPVOID value, int element )
{
 ELEMENTSTRUCT e;
 e.Value = Value;
 e.NumElems = 1;                                /* always 1 */
 e.Element[0].Type = PTYPE_SHORT;               /* always PTYPE_SHORT */
 e.Element[0].Index = element;
 return VBXGetProp( control, index, (LONG) &e );
}

BOOL VBXGetArrayPropByName( HCTL control, LPSTR name,
                    LPVOID value, int element )
{
 return VBXGetArrayProp( control,
   VBXGetPropIndex( control, name ), value, element );
}

BOOL VBXSetArrayProp( HCTL control, int index,
                LONG value, int element )
{
 ELEMENTSTRUCT e;
 e.Value = Value;
 e.NumElems = 1;                                /* always 1 */
 e.Element[0].Type = PTYPE_SHORT;               /* always PTYPE_SHORT */
 e.Element[0].Index = element;
 return VBXSetProp( control, index, (LONG) &e );
}

BOOL VBXSetArrayPropByName( HCTL control, LPSTR name,
                    LONG value, int element )
{
 return VBXSetArrayProp( control,
   VBXGetPropIndex( control, name ), value, element );
}
```

12. Form Files

These functions are for use with the header files generated by VBXGEN:

HFORMFILE VBXCreateFormFile(LONG len, LPVOID data)

This function creates a temporary form file from a buffer <data> of <len> bytes of data. The form file returned can be used as an argument to the VBXCreate() function. It returns -1 if an error occurs.

BOOL VBXDeleteFormFile(HFORMFILE file)

This function deletes the form file <file> and frees any resources associate with it. It returns TRUE if successful, or FALSE if an error occurs.

13. 32-bit Issues

TVbxEventHandler as a base class

ObjectWindows windows and dialogs, including those generated by AppExpert, which use VBX controls must have TVbxEventHandler as a base class if built as a 32-bit application. For example:

```
class TMyDialog : public TDialog
...
DEFINE_RESPONSE_TABLE1(TMyDialog, TDialog)
```

should be

```
class TMyDialog : public TDialog, public TVbxEventHandler
...
DEFINE_RESPONSE_TABLE2(TMyDialog, TDialog, TVbxEventHandler)
```

This can done manually, as shown above, or with ClassExpert. To make this modification using ClassExpert, perform the following steps:

Select the target in the project window
Select View | Class Expert
Select the desired window or dialog class in the Classes window
Select the Control Notifications item in the Events window
Select a VBX control under the Control Notifications item
Select a VBX event under the VBX control item
Use a local menu to add a handler for the selected event
ClassExpert will make sure that the window or dialog class is derived from TVbxEventHandler

If TVbxEventHandler is not used as a base class, the VBX control will not appear.

Choosing Data Types

It's important to use the correct data type when getting property values from a VBX control. This is not always obvious. For example, the following code looks quite reasonable and works in 16-bit:

```
int count;
VBXGetPropByName( hCtl, "Count", &count );
```

This same code will not work in 32-bit since <count> is now 32-bits wide and the emulator (which is 16-bit) only writes 16-bits of information. The lower 2 bytes of <count> are left uninitialized. The best way to fix this is to make <count> a short:

```
 short count;
 VBXGetPropByName( hCtl, "Count", &count );
```

Please refer to the table in Section 4 for appropriate data types.

Windows NT

VBX events are not forwarded to ObjectWindows child objects under Windows NT. They are, however, correctly forwarded to the parent object.