

# VCI Wrapper

## COM Object for VCI

---

## **IXXAT**

### **Headquarter**

IXXAT Automation GmbH  
Leibnizstr. 15  
D-88250 Weingarten

Tel.: +49 (0)7 51 / 5 61 46-0  
Fax: +49 (0)7 51 / 5 61 46-29  
Internet: [www.ixxat.de](http://www.ixxat.de)  
e-Mail: [info@ixxat.de](mailto:info@ixxat.de)

### **US Sales Office**

IXXAT Inc.  
120 Bedford Center Road  
USA-Bedford, NH 03110

Phone: +1-603-471-0800  
Fax: +1-603-471-0880  
Internet: [www.ixxat.com](http://www.ixxat.com)  
e-Mail: [sales@ixxat.com](mailto:sales@ixxat.com)

## **Support**

In case of unsolvable problems with this product or other IXXAT products please contact IXXAT in written form by:

Fax: +49 (0)7 51 / 5 61 46-29  
e-Mail: [support@ixxat.de](mailto:support@ixxat.de)

## **Copyright**

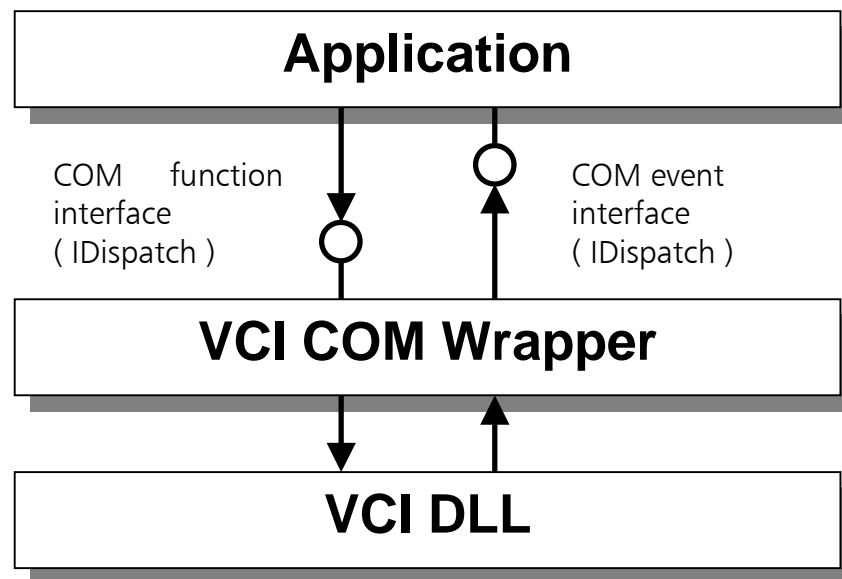
Duplication (copying, printing, microfilm or other forms) and the electronic distribution of this document is only allowed with explicit permission of IXXAT Automation GmbH. IXXAT Automation GmbH reserves the right to change technical data without prior announcement. The general business conditions and the regulations of the license agreement do apply. All rights are reserved.

<b>1</b>	<b>INTRODUCTION .....</b>	<b>5</b>
1.1	Field of application.....	5
1.2	Installation.....	5
<b>2</b>	<b>PROGRAMMING INFORMATION .....</b>	<b>6</b>
2.1	Description of functions.....	6
2.2	Initialization (Visual Basic).....	7
2.3	Event Handler (Visual Basic) .....	9
2.4	Restrictions .....	10
2.5	Example programs.....	11



# 1 Introduction

The VCI Wrapper software component is based on the COM (Component Object Model) technology and discloses the functionality of the VCI DLL via automation-compatible interfaces (see VCI-handbook).



Applications which connect directly to the VCI-DLL use so-called callback handlers for the signaling of asynchronous events. When using the VCI Wrapper software component, such events are passed on to the application via COM events.

## 1.1 Field of application

Since the VCI-DLL signals asynchronous events via callback handlers, the VCI DLL must be given function addresses. Since this is not possible in Visual Basic 6.0, or only possible to a limited extent, a way was found with the VCI Wrapper component to write VCI applications under Visual Basic 6.0.

As already mentioned, the VCI Wrapper component discloses its functionality via automation-compatible interfaces and is thus ideally suited to Visual Basic.

## 1.2 Installation

The VCI Wrapper component is installed with the VCI V2.14 (or higher). Therefore no separate installation is necessary.

---

## 2 Programming information

The VCI Wrapper component offers almost the same functionality as the VCI DLL. In this section, therefore, only the differences or the special features for programming under Visual Basic are discussed. In the following descriptions, therefore, it is assumed that you have already worked through the Programmer Manual of the Virtual CAN Interface.

### 2.1 Description of functions

The VCI Wrapper component supports all VCI functions with the following exceptions or restrictions:

- No structures can be returned via the automation interface (IDispatch). With VCI-functions which have structures as output parameters in their original form, each structure element becomes its own output parameter. This category comprises the following functions:

VCI\_ReadBoardInfo  
VCI\_ReadBoardStatus  
VCI\_ReadCanInfo  
VCI\_ReadCanStatus  
VCI\_ReadQueObj

Example:

```
typedef struct{
    UINT8 sts;
    UINT8 cpu_load;
}VCI_BRD_STS;

int VCI_ReadBoardStatus( UINT16 board_hdl , VCI_BRD_STS* p_sts );
```



```
HRESULT VCI_ReadBoardStatus( [in] long board_hdl ,
                             [out] unsigned char* status ,
                             [out] unsigned char* cpu_load ,
                             [out,retval] int* ret_code) ;
```

- The VCI Wrapper component does not support any callback handlers. Therefore, with the function **VCI\_PrepareBoard**, the parameters for information on callback handlers are missing.

- The following functions are not supported by the VCI component:

VCI\_PrepareBoardMsg  
VCI\_PrepareBoardVisBas  
VCI\_TestBoard

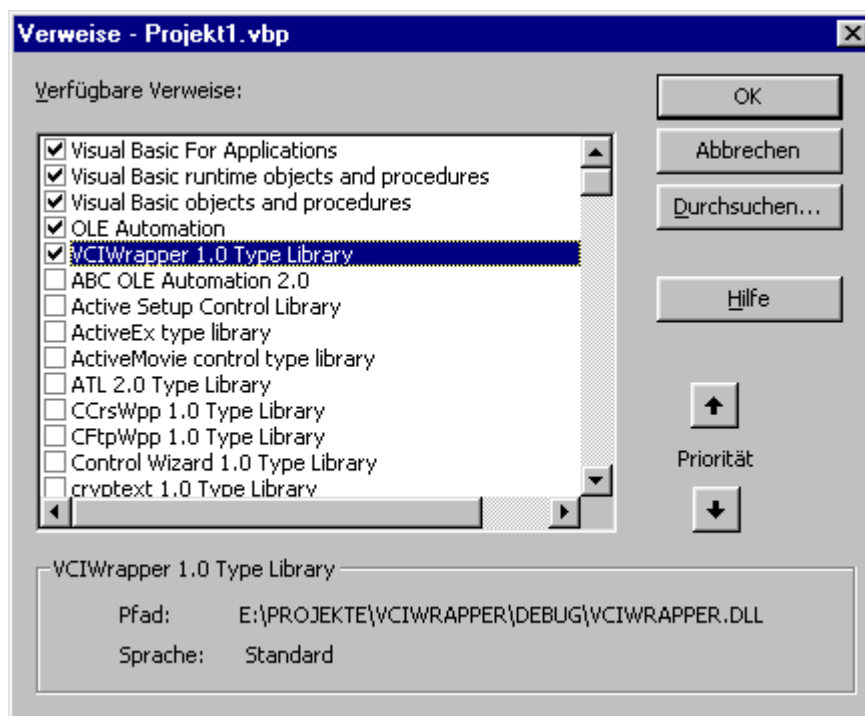
- The VCI-Wrapper additionally supports a board information collection, which provides detailed info about installed CAN-hardware (VCI Version 2.16 or higher).

## 2.2 Initialization (Visual Basic)

This section shows step by step how to integrate the VCI Wrapper component into your Visual Basic 6.0 project and initialize it.



1. Create the Visual Basic 6.0 project and select in the **Reference-Dialog** field the Type Library of the VCI Wrapper component. The Reference-Dialog field can be accessed via the menu point **Project/References...**



- 
2. Insert the module **Vci2.bas** into your Visual Basic project. Vci2.bas contains symbolic constants which make your code more readable.



3. In the second step the definition of an object-variable for the VCI Wrapper component must be inserted into your Visual Basic module. This declaration should have the following appearance:

**Dim WithEvents objVCI As VCIWRAPPERLib.CVCIWrap**

It is important not to omit the key-word `WithEvents` here, as otherwise no COM-Events can be received. In the above example `objVCI` is a freely selectable variable name – however, the reference `VCIWRAPPERLib.CVCIWrap` must be entered in exactly this form.

4. Before the VCI Wrapper component can be accessed, the aforementioned object-variable must be initialized. This could be done, for example, in the load-handler of the Visual Basic form. For this, the following instruction is necessary:

**Set objVCI = New VCIWRAPPERLib.CVCIWrap**

5. After successful initialization of the object variables, the methods of the VCI Wrapper component are available:

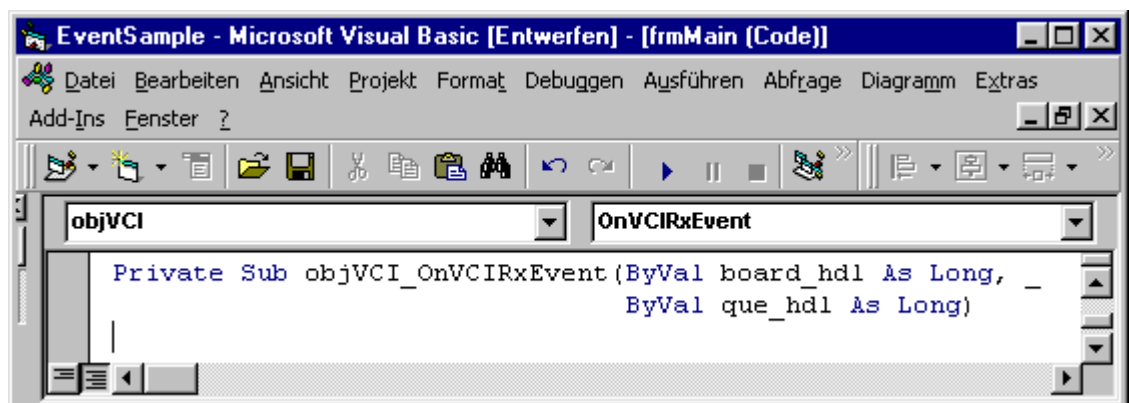
e.g.: `intBoardHandle = objVCI.PrepareBoard( VCI_IPCI165, &HDC00, 9 )`

### 2.3 Event Handler (Visual Basic)

The VCI Wrapper component signals reception of CAN messages via COM events.



In order for these COM events to be processed in the application, a corresponding event-handler must be present. Insert this event-handler into your application by selecting the VCI Wrapper component in the object-list (in this example objVCI) and then the element **OnVCIRxEvent**.



By means of this procedure the appropriate event-handler is automatically inserted into your application:

```
Private Sub objVCI_OnVCIRxEvent (    ByVal board_hdl As Long,
                                   ByVal que_hdl As Long )
End Sub
```

With the information supplied by the event-procedure (Board Handle and Queue Handle) the received CAN-messages can be read out of the corresponding queue with the function `VCI_ReadQueObj`.



**In the event handler, the queues should be read out as quickly as possible.**

---

## 2.4 Reading board information

Till VCI 2.16 the VCI-Wrapper supports a board collection object to retrieve information about installed CAN-Hardware. To use this kind of object you have to add the following declaration to your VB-Program:

```
Dim objBoardEnum As VCIWRAPPERLib.BoardCollection
```

This object implements a collection of board info objects. You can traverse them with the „for each“ statement:

```
Dim board As VCIWRAPPERLib.IBoardInfo
For Each board In objBoardEnum
    lbBoards.AddItem board.Name & vbTab & board.Info
Next
```

The collection also supports the „Item“ and „Count“ properties which can be used to enumerate the collection items:

```
Dim iCount As Integer
For iCount = 1 To objBoardEnum.Count
    lbBoards.AddItem objBoardEnum.Item(iCount).Name
Next
```

Via the „DefaultBoard“ property you are able to determine the default board specified in the control panel applet. If no default board has been specified it returns the first available board.

A boardInfo object has the following properties:

<b>Type</b>	VCI-internal board type, used for the initialisation via VCI2_Prepareboard.
<b>Index</b>	Board-Index (Hardware-Key), , used for the initialisation via VCI2_Prepareboard.
<b>Position</b>	board type dependant position (e.g. second iPCI 165 PCI)
<b>Name</b>	board name (e.g. iPCI 165 PCI)
<b>Manufacturer</b>	manufacturer name (e.g. IXXAT Automation GmbH)
<b>Info</b>	board information (e.g. IRQ, base address)
<b>AdditionalInfo</b>	additional info, used for the initialisation via

VCI2_Prepareboard. (e.g.. IP-Adresse of <a href="#">CAN@NET</a> )
---

A board info object could be used to initialise a VCI board as follows:

```
Dim board As IVCIWrap_BoardInfo
Set board = objBoardEnum.Item(iSelected)
iRet = objVCI.VCI2_PrepareBoard(board.Type, _
                                board.Index, _
                                board.AdditionalInfo)
```

## 2.5 Restrictions

Generally performance critical applications should fall back on VCI DLL (that is not possible with Visual Basic 6.0!), since due to the VCI Wrapper component an additional software layer must be run through between hardware and application.

The VCI Wrapper component is a so-called Apartment Threaded component and is therefore ideally integrated into the architecture of Visual Basic. This means that Receive COM events are passed on to the application via Windows messages. For the Visual Basic application it is therefore extremely important that the Windows Message Queue is continually processed.



With the activation of a modal dialog, the above-mentioned condition is no longer fulfilled. While the modal is open, no Receive events can be processed and CAN-messages can be lost. In addition, long calculations etc should be avoided in all event handlers of your Visual Basic application, as this also slows down the processing of the Windows Message Queue.

In the VCI Wrapper component there are additional queues for the intermediate storage of received CAN-messages. The size of these queues is 3000 messages. This means that with delayed read-out, 3000 messages can be temporarily stored without messages being lost. If the queue is full, and there is no more room for a further message, the last message in the queue is marked with the Queue Overrun status-code 0x80 (see also VCI Programmer Manual)

## 2.6 Example programs

The VCI Wrapper component contains two Visual Basic examples which demonstrate the transmission and reception of CAN-messages. Reading queues event controlled and via polling is shown.