

# Visual Clipper via GTWVW

---

**WVWClip Documentation**

Copyright 2016-2026 Ashfaq Sial

## Contents

Background .....	3
Build Instructions .....	6
@...BOX .....	7
@...GET .....	8
@...GET CHECKBOX .....	9
@...GET EDITBOX.....	10
@...GET LISTBOX .....	11
@...GET PASSWORD .....	12
@...GET PUSHBUTTON .....	13
@...GET RADIOGROUP .....	14
@...GET TBROWSE.....	15
@...IMAGE.....	17
@...LABEL .....	18
@...LABELOBJ .....	19
@...SAY .....	20
@...SAY PUSHBUTTON.....	21
WAlert().....	22
WMENU TO .....	23
WApp() Class .....	24
:WClose().....	25
:WCLS().....	26
:WinInit() .....	27
:WOpen().....	28
:WPopup() .....	29
Other useful functions in this library .....	30
Modifications for Harbour 3.2dev GTWVW Sources .....	31

## Background

I began with the following Clipper 5.3 code. It functions as expected when compiled using Harbour.

```
#include "inkey.ch"

PROCEDURE main()

    LOCAL cName      := 'Fred Bloggs Jr.'
    LOCAL dDOB       := Date() - 20000
    LOCAL cStatus    := 'Single'
    LOCAL nSalary    := 95000
    LOCAL lEmployed  := .T.
    LOCAL lOK        := ''
    LOCAL lCancel    := ''

    SET CENTURY ON
    SetMode( 10, 32 )
    CLS

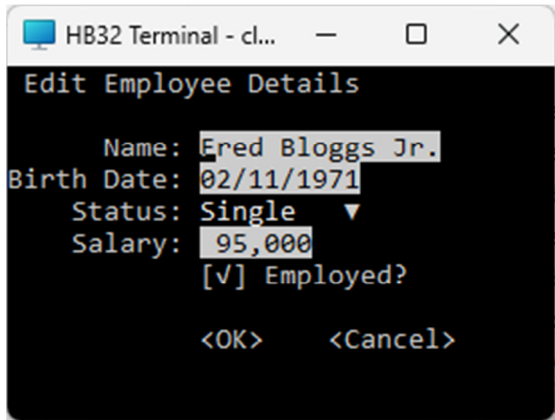
    @ 00, 01 SAY 'Edit Employee Details'
    @ 02, 12 GET cName      CAPTION 'Name:'          PICTURE '@K'
    @ 03, 12 GET dDOB       CAPTION 'Birth Date:'    PICTURE '@D'
    @ 04, 12, 08, 21 GET cStatus LISTBOX { 'Unknown', 'Married', 'Single' } ;
        DROPDOWN CAPTION 'Status:'
    @ 05, 12 GET nSalary    CAPTION 'Salary:'        PICTURE '999,999'
    @ 06, 12 GET lEmployed CHECKBOX CAPTION 'Employed?'
    @ 08, 12 GET lOK       PUSHBUTTON CAPTION 'OK' ;
        STATE {|| hb_keyPut( K_CTRL_W ) }
    @ 08, 20 GET lCancel   PUSHBUTTON CAPTION 'Cancel' ;
        STATE {|| hb_keyPut( K_ESC ) }

    READ

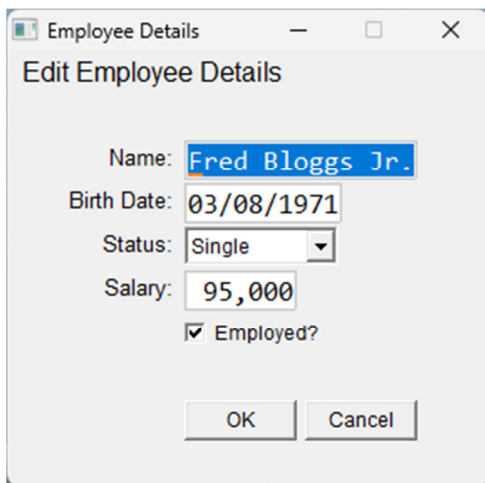
    IF LastKey() <> K_ESC
        Alert( cName + ';' + ;
            DToC( dDOB ) + ';' + ;
            cStatus + ';' + ;
            Str( nSalary ) + ';' + ;
            iif( lEmployed, 'Y', 'N' ) )
    ENDIF

    RETURN
```

The resulting output looks like this.



After making a few changes, highlighted in code shown below, the output looks much better.



```
#include "inkey.ch"
#include "wvclip.ch"
```

```
PROCEDURE main()
```

```
LOCAL cName      := 'Fred Bloggs Jr.'
LOCAL dDOB       := Date() - 20000
LOCAL cStatus    := 'Single'
LOCAL nSalary    := 95000
LOCAL lEmployed  := .T.
LOCAL lOK        := ''
LOCAL lCancel    := ''
```

```
SET CENTURY ON
SetMode( 10, 32 )
```

```
PUBLIC oApp := WApp():New( .T., , 'Employee Details' )
```

```
@ 00, 01 LABEL 'Edit Employee Details' FONT HeadFont()
@ 02, 12 GET cName      CAPTION 'Name:'      PICTURE '@K'
@ 03, 12 GET dDOB      CAPTION 'Birth Date:' PICTURE '@D'
@ 04, 12, 08, 21 GET cStatus LISTBOX { 'Unknown', 'Married', 'Single' } ;
    DROPDOWN CAPTION 'Status:'
@ 05, 12 GET nSalary   CAPTION 'Salary:'     PICTURE '999,999'
@ 06, 12 GET lEmployed CHECKBOX CAPTION 'Employed?'
@ 08, 12 GET lOK      PUSHBUTTON CAPTION 'OK' ;
    STATE {|| hb_keyPut( K_CTRL_W ) }
@ 08, 20 GET lCancel  PUSHBUTTON CAPTION 'Cancel' ;
    STATE {|| hb_keyPut( K_ESC ) }
```

```
READ
```

```
IF LastKey() <> K_ESC
    WAlert( cName + ';' + ;
        DToC( dDOB ) + ';' + ;
        cStatus + ';' + ;
        Str( nSalary ) + ';' + ;
        iif( lEmployed, 'Y', 'N' ) )
ENDIF
```

```
RETURN
```

## Build Instructions

Adjust the code as noted above. See wclip.prg and wclip.hbp files in **tests** folder.

Compile.

```
Hbmk2 wclip.hbp
```

### Notes:

1. *This library was built using Harbour 3.2dev with some modifications to the accompanying GTWVW sources as described at the end of this document.*
2. *Source code contains detailed information regarding each element of this library.*

## @...BOX

Draw a box at given coordinates.

Syntax

```
@ <nTop>, <nLeft>, <nBottom>, <nRight> BOX  
    RAISED|RECESSED|GROUP;  
    [CAPTION <cCaption>] ;  
    [OFFSET<aTLBR>]
```

Arguments

<cCaption>  
Caption text.

<aTLBR>  
This is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

Example

```
@ 0, 0, MaxRow(), MaxCol() BOX RECESSED  
  
@ 4, 23, 7, 33 BOX GROUP ;  
    CAPTION 'Database'
```

## @...GET

Draw a data entry field at given coordinates.

Syntax

```
@ <nRow>, <nCol> GET <v> ;  
    [CAPTION <cCaption>] ;  
    [PICTURE <cPicture>] ;  
    [VALID <valid>] [WHEN <when>] ;  
    [SEND <msg>]
```

Example

```
LOCAL nSalary := 95000
```

```
@ 05, 12 GET nSalary CAPTION 'Salary:' PICTURE '999,999';  
    VALID nSalary >= 0
```

## @...GET CHECKBOX

Draw a checkbox at given coordinates.

Syntax

```
@ <nRow>, <nCol> GET <lVar> CHECKBOX ;  
    [CAPTION <cCaption>];  
    [VALID <valid>] [WHEN <when>] ;  
    [OFFSET <aTLBR>
```

Arguments

Same as those for Clipper except for the additional ones below.

<aTLBR>

This is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

Example

```
LOCAL lEmployed := .T.
```

```
@ 06, 12 GET lEmployed CHECKBOX ;  
    CAPTION 'Employed?'
```

## @...GET EDITBOX

Draw an edit box at given coordinates.

Syntax

```
@ <nTop>, <nLeft>, <nBottom>, <nRight> GET <cVar> EDITBOX ;  
    [CAPTION <cCaption>] [FONT <aFont> ;  
    [STYLE <nStyle>] [MAXCHAR <nMaxChar>] ;  
    [OFFSET <aTLBR>]
```

Arguments

<nTop>, <nLeft>, <nBottom>, <nRight>

Edit box coordinates.

<cVar>

This is the edited variable initialized as a character string.

<cCaption>

Caption text. It is drawn so that it ends at <nLeft>-1 column position.

<aFont>

Array of two elements: {cFont, nSize}, specifies which font will be used for this edit box.

<nStyle>

This could be any combination of ES\_\* constants. See example below.

<nMaxChar>

Maximum number of input characters accepted.

<aTLBR>

This is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

Example

```
LOCAL cMemo := 'These are employee notes.'
```

```
@ 0, 5, 6, 35 GET cMemo EDITBOX CAPTION 'Notes:' ;  
    STYLE + WS_VSCROLL + WS_HSCROLL + ES_MULTILINE + ES_READONLY
```

## @...GET LISTBOX

Draw a drop down type list box at given coordinates. Behaviour is similar to standard windows convention (i.e., ENTER/ESC will kill focus from listbox).

### Syntax

```
@ <nTop>, <nLeft>, <nBottom>, <nRight> GET <nVar|cVar> LISTBOX <aOptions> ;  
    [DROPDOWN] ;  
    [VALID <valid>] [WHEN <when>] ;  
    [CAPTION <cCaption>] ;  
    [OFFSET <aTLBR>]
```

### Arguments

<nTop>, <nLeft>, <nBottom>, <nRight>  
LISTBOX coordinates.

<nVar|cVar>  
This is the edited variable initialized as a numeric value or a character string.

If nVar, it is assigned the index (first item is 1) of the selected item of <aOptions>.

If cVar, it is assigned the string of the selected item of <aOptions>.

<aOptions>  
An array of options: Each option length must be > 2.

<cCaption>  
Caption text. It is drawn so that it ends at <nLeft>-1 column position.

<aTLBR>  
This is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

### Example

```
LOCAL cStatus := 'Single'  
  
@ 04, 12, 04, 19 GET cStatus LISTBOX {'Unknown', 'Married', 'Single'} ;  
    CAPTION 'Status:'
```

## @...GET PASSWORD

Draw a password entry field at given coordinates.

Syntax

```
@ <nRow>, <nCol> GET <v> PASSWORD;  
    [CAPTION <cCaption>] ;  
    [PICTURE <cPicture>] ;  
    [VALID <valid>] [WHEN <when>] ;  
    [SEND <msg>]
```

Arguments

Example

```
PRIVATE cPWD := Space(8)
```

```
@ 05, 12 GET cPWD PASSWORD CAPTION 'Password:' PICTURE '@K';  
    VALID !Empty( cPWD )
```

## @...GET PUSHBUTTON

Draw a push button at given coordinates.

Syntax

```
@ <nRow>, <nCol> GET <lVar> PUSHBUTTON ;  
    [CAPTION <cCaption>] ;  
    [VALID <valid>] [WHEN <when>] ;  
    [STATE <bAction>] ;  
    [OFFSET <aTLBR>]
```

Arguments

<nRow>, <nCol>  
Pushbutton coordinates.

<lVar>  
This is a placeholder for the underlying GET. It is always set to .T..

<cCaption>  
Caption text is used for button label.

<nWidth>  
Push button width. It defaults to 7.

<bAction>  
This code block is evaluated when pushbutton is clicked by user.

<aTLBR>  
This is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

Example

```
LOCAL lSave := .T.  
LOCAL nBtn := 0  
  
@ 08, 02 GET lSave PUSHBUTTON ;  
    CAPTION 'OK';  
    STATE {|| nBtn := 1, hb_keyPut( K_CTRL_W )}
```

## @...GET RADIOGROUP

Draw a group of radio buttons at given coordinates.

Syntax

```
@ <nTop>, <nLeft>, <nBottom>, <nRight> GET <nVar> RADIOGROUP ;  
    [CAPTION <cCaption>] ;  
    [OFFSET <aTLBR>]
```

Arguments

Same as those for Clipper except for the additional ones below.

<aTLBR>

This is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

Example

```
LOCAL nRB  
LOCAL aRadio[ 3 ]  
  
aRadio[ 1 ] := "Red"  
aRadio[ 2 ] := "Green"  
aRadio[ 3 ] := "Blue"  
  
nRB := 2           // Default radio button  
  
@ 2, 8, 4, 15 GET nRB RADIOGROUP aRadio ;  
    CAPTION "Colour"
```

## @...GET TBROWSE

Display browse object on screen.

### Syntax

```
@ <nTop>, <nLeft>, <nBottom>, <nRight> GET <idVar> ;
    TBROWSE <oBrowse> ;
    [VALID <valid>] [WHEN <when>] ;
    [SEND <msg>] ;
    [GUISEND <guimsg>]
```

### Arguments

Same as those for Clipper version except that MESSAGE clause is not implemented.

### Example

```
#include "inkey.ch"
#include "wwwclip.ch"

REQUEST DBFCDX

FUNCTION Main()

LOCAL lDummy := .F.
LOCAL oBrowse := NIL

PRIVATE GetList := {}
PRIVATE oApp

SET DEFAULT TO .\data
SET EVENTMASK TO INKEY_ALL - INKEY_MOVE

hb_cdpSelect( 255 )

SetMode( 8, 28 )

oApp := WApp():New( .F., , "Browse Example" )

USE currency INDEX currency VIA "DBFCDX" SHARED

// Setup browse.
oBrowse := TBrowse():New()
oBrowse:colorspec := 'W/N, N/W'

// Add column.
oBrowse:AddColumn( TBColumn():New( , { || Pad( currency->currid + ' ' +
    currency->descript + ' ' + currency->symb, 29 ) } ) )
@ 00, 01 LABEL 'Currencies' FONT HeadFont()
@ 02, 01 LABEL 'ID'
```

```
@ 02, 05 LABEL 'Currency'
```

```
@ 02, 21 LABEL 'Symbol'
```

```
@ 03, 01, 06, 26 GET 1Dummy TBROWSE oBrowse GUISEND forceStable()
```

```
READ SAVE
```

```
USE
```

```
oApp:WCLS()
```

```
RETURN NIL
```

## @...IMAGE

Draw an image at given coordinates.

Syntax

```
@ <nTop>, <nLeft>, <nBottom>, <nRight> IMAGE <cFile>;  
    [OFFSET <lTight>|<aTLBR>] TBITMAP
```

Arguments

<nTop>, <nLeft>, <nBottom>, <nRight>  
IMAGE coordinates.

<cFile>  
Image filename or index into an image list.

<lTight>|<aTLBR>  
lTight When set to .T. the image is placed snugly within the image coordinates.

aTLBR is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

<TBITMAP>  
Image is a transparent bitmap.

Example

```
@ 1, 1, 3, 7 IMAGE hb_DirBase() + "logo.bmp" TBITMAP
```

## @...LABEL

Draw a label at given coordinates.

Syntax

```
@ <nRow>, <nCol> LABEL <cLabel>;  
    [FONT <aFont>] ;  
    [RIGHT] ;  
    [COLOR <cClr>]
```

Arguments

<nRow>, <nCol>  
Label coordinates.

<cLabel>  
Label text.

<aFont>  
Array of three elements: {cFont, nSize, nWeight }. Specifies which font will be used to draw the label with.

Default is "Arial", 16, FW\_NORMAL.

RIGHT  
If specified, the expression output will end at <nCol> position.

<cClr>  
Output colour is in 'N/W' format.

Example

```
LOCAL aFont := {'Arial', 15, FW_NORMAL }  
  
@ 02, 05 LABEL 'Use at your own risk.';  
    FONT aFont
```

## @...LABELOBJ

Draw a label object at given coordinates.

Syntax

```
@ <nTop>, <nLeft>[, <nBottom>[, <nRight>]] LABELOBJ <cLabel>;  
    [WIDTH <nWidth>] [RIGHT] ;  
    [FONT <aFont>] ;  
    [OFFSET <aTLBR>] ;  
    [COLOR <cClr>]
```

Arguments

<nTop>, <nLeft>, <nBottom>, <nRight>  
Label coordinates.

<cLabel>  
Label text.

<nWidth>  
Label width.

RIGHT  
If specified, the expression output will be right justified.

<aFont>  
Array of three elements: {cFont, nSize, nWeight}, specifies which font will be used to draw the label with.

Default is {"Arial", 16, FW\_NORMAL}.

<aTLBR>  
This is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

<cClr>  
Output colour in 'N/W' format.

Example

```
LOCAL aFont := {'Arial', 17, FW_BOLD }  
@ 02, 05, 03, 21 LABELOBJ 'Employee Details' WIDTH 25 RIGHT FONT aFont
```

## @...SAY

Output value of an expression at given coordinates.

### Syntax

```
@ <nRow>, <nCol> SAY <exp> ;  
    [CAPTION <cCaption>] ;  
    [PICTURE <cPicture>] ;  
    [COLOR <cColorString>]
```

### Arguments

All arguments are the same as Clipper 5.3 except the additional ones described below.

#### <cCaption>

Text label for <exp>.

### Example

```
@ 01, 10 SAY 'My Application V10.0' ;  
    CAPTION 'Application:'
```

### Output

Application: My Application V10.0

## @...SAY PUSHBUTTON

Draw a push button at given coordinates. This pushbutton will not be part of GET subsystem.

Syntax

```
@ <nRow>, <nCol> SAY PUSHBUTTON ;  
    [CAPTION <cCaption>] ;  
    [WIDTH <nWidth>] ;  
    [STATE <bAction>] ;  
    [OFFSET <aTLBR>]
```

Arguments

<nRow>, <nCol>

PUSHBUTTON coordinates. Pushbutton is 7 characters wide.

<cCaption>

Caption text is used for button label.

<nWidth>

Button width. It defaults to 7.

<bAction>

This code block is evaluated when pushbutton is clicked by user.

<aTLBR>

This is an array of four elements, {nTop, nLeft, nBottom, nRight} in pixels, used for row and column alignment.

Example

```
@ 08, 02 SAY PUSHBUTTON ;  
    CAPTION 'OK' ;  
    STATE {|| hb_keyPut( K_CTRL_W )}
```

## **WAlert()**

This function is a GUI version of Alert() but only with OK button.

Syntax

```
WAlert( <cMsg>[, <cHead>] )
```

Arguments

<cMsg >

Text message to be displayed. Semicolon is used to fold the text message over multiple lines.

<cHead>

Heading text. It defaults to 'Alert'.

Example

```
WAlert( cName + ';' + ;  
        DToC( dDOB ) + ';' + ;  
        cStatus + ';' + ;  
        Str( nSalary ) + ';' + ;  
        iif( lEmployed, 'Y', 'N' ) )
```

## **WMENU TO**

Returns last menu event of an active menu. See WVWMENU.PRG for more details.

### Syntax

```
WMENU TO <nVar>
```

### Arguments

<nVar>

The variable receives the last menu event of active menu.

### Example

```
WMENU TO nOpt
```

## WApp() Class

Setup an application (main) window using GTWVW environment. This method is only called once at the start of an application. See GTWVW documentation.

### Syntax

```
WApp():New( <lGUIApp>, <cIconFile >, <cTitle >, <aTBBtns > )  
    => oApp
```

### Arguments

#### <lGUIApp>

Set it to .T. for GUI application. Default .F. - Console application.  
Line spacing is set to 6 (GUI application) or 0 (Console application).

#### <cIconFile>

Application icon file.

#### <cTitle>

Main window's title.

#### <aTBBtns>

An array of buttons for the main window's toolbar.

### Example

```
PUBLIC oApp
```

```
oApp := WApp():New( .T., hb_DirBase() + "resource\wwclip.ico", "My Application" )
```

### Methods

```
oApp:WinInit( cTitle, lGUIWin, lCentre )  
oApp:WOpen( nTop, nLeft, nBottom, nRight, cTitle, lGUIWin, lCentre )  
oApp:WPopup( nTop, nLeft, nBottom, nRight, lGUIWin, lCentre )  
oApp:WClose()  
oApp:WCLS()  
oApp:EnableMenu( nNumItem )           Enable main menu  
oApp:DisableMenu( nNumItem )         Disable main menu  
oApp:AddTBBButtons()                 Add buttons to main toolbar  
oApp:ClrTBBButtons()                 Clear buttons from main toolbar  
oApp:CreateToolbar()                  Create empty main toolbar  
oApp:DestroyToolbar()                 Destroy main toolbar  
oApp:DisableToolbar()                 Disable main toolbar  
oApp:EnableToolbar()                  Enable main toolbar  
oApp:SBCreate( aParts )               Create main status bar with an array of parts  
oApp:SBSSetText( nPartno, cText )     Place text in a given part of main status bar
```

## **:WClose()**

Close topmost window.

Syntax

oApp:WClose()

Arguments

None

## **:WCLS()**

Clear topmost window, Get List and associated GUI objects.

Syntax

oApp:WCLS()

Arguments

None

## **:WinInit()**

Initialize current window and its GUI elements and then display it.

Syntax

```
oApp:WinInit( [<cTitle>], [<lGUIWin>], [lCentre] )
```

Arguments

<cTitle>

Window's heading.

<lGUIWin>

.T. – GUI window (GETs will have boxes drawn around them)

.F. – Console window

Defaults to application type specified in WApp() call.

<lCentre>

Flag to centre the window. Defaults to .F..

Example

```
oApp:WinInit( 'Edit Employee Details', .T., .T. )
```

## **:WOpen()**

Open a new child window.

Syntax

```
oApp:WOpen( <nTop>, <nLeft, <nBottom>, <nRight>, ;  
            <cTitle>, <lGUIWin>, <lCentre> ) ;  
=> nWinNum
```

Arguments

<nTop>, <nLeft, <nBottom>, <nRight>

Child window coordinates

<cTitle>

Window's title.

<lGUIWin>

.T. – GUI window (GETs will have boxes drawn around them)

.F. – Console window

Defaults to application type specified in WApp() call.

<lCentre>

Flag to centre the window. Defaults to .F..

Example

LOCAL nWinNum

```
nWinNum := oApp:WOpen( 2, 3, 11, 35, 'Browse Example', .T., .T. )
```

## **:WPopup()**

Open a new popup window. These windows are created with WS\_POPUPWINDOW style.

Syntax

```
oApp:WPopup( <nTop>, <nLeft, <nBottom>, <nRight> ;  
            <lGUIWin>, <lCentre> )  
=> nWinNum
```

Arguments

<nTop>, <nLeft, <nBottom>, <nRight>

Popup window coordinates

<lGUIWin>

.T. – GUI window (GETs will have boxes drawn around them)

.F. – Console window

Defaults to application type specified in WApp() call.

<lCentre>

Flag to centre the window. Defaults to .F..

Example

```
LOCAL nWinNum
```

```
nWinNum := oApp:WPopup( 2, 3, 11, 35 )
```

## Other useful functions in this library

```
AddGuiObj( nWinNum, bAction )  
ClrGuiObjs( nWinNum )
```

```
LenGuiObjs( nWinNum )  
ReSizeGuiObjs( nWinNum, aLen )
```

```
AddGuiCtl( nWinNum, bAction )  
ClrGuiCtls( nWinNum )
```

```
ColumnFont()  
HeadFont()  
MenuFont()  
LabelFont()  
LabelFontSize()
```

## Modifications for Harbour 3.2dev GTWVW Sources

Modify GTWVW sources, as shown below, to work with Harbour 3.2.0dev. Rebuild GTWVW library.

### gtwvwd.c

Comment out the following two lines.

```
1238: hb_wvw_vmouse_Init()  
1247: hb_wvw_vmouse_Exit()
```

### wwwutils.c

Replace line 435

```
BS_AUTORADIOBUTTON /*| WS_GROUP*/ );
```

with

```
BS_AUTORADIOBUTTON | hb_parni( 13 ) /* nStyle */ );
```

13th parameter - when set to .T., it makes a radio button the first of a group of radio buttons.

Add this function to the end of wwwutils.c

```
HB_FUNC( WWV_GETCTRLHANDLE )  
{  
    UINT usWinNum = WWV_WHICH_WINDOW;  
    BYTE  bStyle = 0;  
  
    hb_retnl( (LONG) FindControlHandle(usWinNum,  
                                       (BYTE) hb_parni(2),  
                                       (UINT) hb_parni(3),  
                                       &bStyle) );  
  
    hb_storni( bStyle, 4 );  
}
```