Bos Taurus

Graphic Library for

HMG

Version 1.0.5

(c) Dr. Claudio Soto

srvet@adinet.com.uy

http://srvet.blogspot.com

Uruguay, 2012 - 2014

CONTENT

1) *What is **HMG?***

Harbour-MiniGUI (HMG) is a GUI free open source library for Windows developed for the **Harbour** compiler. Harbour is a cross-platform free open source compiler for the xBase language (http://harbour-project.sourceforge.net), is 100% compatible with Clipper language.

There are several teams of developers Harbour-GUI libraries, the two most important free libraries are **HMG Official** (www.hmgforum.com) and **HMG Extended** (http://tech.groups.yahoo.com/group/harbourminigui). In this text we will use the acronym **HMG** to generically refer to both versions of HMG (Official and Extended) interchangeably.

2) *What is the graphics library **Bos Taurus?***

Bos Taurus is a set of graphics functions (free open code) written in C and Harbour based in the Windows GDI functions specially developed for easy programming of the **ON PAINT** events in **HMG**.

In the past, the ON PAINT event was little used in applications developed in HMG due to malfunction of this event, but recently since version 3.0.43 of **HMG Official** (2012/08/30) and version 2.1.5 of **HMG Extended** (2012/09/12) the bug was corrected.

The ON PAINT event runs in response to WM_PAINT message that the Windows System sent to a window for inform that have to paint the client area. In HMG the ON PAINT event allows paint directly on the client area of a window before paint the Controls (Button, Grid, Image, ComboBox, etc.) and before paint the Draw Commands (Draw Graph, Rectangle, Line, etc.).

The **BosTaurus_ChangeLog.TXT** contains the latest updates in the source files of the library. From version 1.0.3, **Bos Taurus** supports **ANSI and UNICODE** text formats.

3) *Why use the graphics library **Bos Taurus**?*

Because in HMG exist very few native functions to draw and manipulate graphic images.

In **Bos Taurus** the draw functions are called with a **Handle Device Context (hDC)** obtained from the selected output device (desktop, client area, bitmap, etc.) just like when programming in C for Windows. This allows the use of the same functions of drawing and the easy transfer of graphics from one device to another irrespective of the output device selected. For example if a drawing function we pass to a hDC of window, the function draw in the client area of the window, however, if we pass the hDC associated with a Bitmap, the function draw in the Bitmap image.

The system of use of **hDC** in the functions will help other people to create other drawing functions fully compatible with the graphics library **Bos Taurus**, the programmer only have to use the appropriate hDC that gives the graphics library **Bos Taurus**. In fact with this system can be implemented in Harbour virtually all graphics functions (GDI) of Windows thereby providing a great power of graphic manipulation to the programmer.

4) *How to use the graphics library **Bos Taurus**?*

It is very easy to use the graphics library **Bos Taurus** in HMG because it is part of the library of **HMG Official** and **HMG Extended**.

5) *Prototype Example*

This example is very simple. Loads a Bitmap to the start the application and puts the background image, before drawing the image paints a vertical colors gradient from white to black. When you close the application frees of the memory the handle of the Bitmap. **For more information see the demos that accompanying the Bos Taurus**.

A) HMG Official

```
#include "HMG.CH"

FUNCTION MAIN
PRIVATE hBitmap := 0

DEFINE WINDOW Win1;
AT 0,0;
WIDTH 700;
HEIGHT 600;
TITLE "Prototype Demo";
MAIN;
ON INIT    Proc_ON_INIT ();
ON RELEASE Proc_ON_RELEASE ();
ON PAINT   Proc_ON_PAINT ()

            // Definition of Application Controls
            @ 435, 280 BUTTON Button1 CAPTION "Click";
            ACTION MsgInfo ("Hello")

END WINDOW
CENTER WINDOW Win1
ACTIVATE WINDOW Win1
RETURN

PROCEDURE Proc_ON_INIT
   hBitmap := BT_BitmapLoadFile ("HMG.bmp")
RETURN

PROCEDURE Proc_ON_RELEASE
   BT_BitmapRelease (hBitmap)
RETURN

PROCEDURE Proc_ON_PAINT
LOCAL  hDC, BTstruct
   hDC := BT_CreateDC ("Win1", BT_HDC_PAINTCLIENTAREA, @BTstruct)
        BT_DrawGradientFillVertical (hDC,;
                    0, 0,;
                    BT_ClientAreaWidth ("Win1"),;
                    BT_ClientAreaHeight("Win1"),;
```

5

```
                        WHITE, BLACK)
        BT_DrawBitmap (hDC, 35, 200, 300, 250, BT_COPY, hBitmap)
    BT_DeleteDC (BTstruct)
RETURN
```

B) _HMG Extended_

```
#include "HMG.CH"
#include "BosTaurus.CH"

FUNCTION MAIN
PRIVATE hBitmap := 0

DEFINE WINDOW Win1;
AT 0,0;
WIDTH 700;
HEIGHT 600;
TITLE "Prototype Demo";
MAIN;
ON INIT      Proc_ON_INIT ();
ON RELEASE   Proc_ON_RELEASE ();
ON PAINT     Proc_ON_PAINT ()
ON SIZE      BT_ClientAreaInvalidateAll ("Win1");
ON MAXIMIZE  BT_ClientAreaInvalidateAll ("Win1")

            // Definition of Application Controls
            @ 435, 280 BUTTON Button1 CAPTION "Click";
            ACTION MsgInfo ("Hello")

END WINDOW
CENTER WINDOW Win1
ACTIVATE WINDOW Win1
RETURN


PROCEDURE Proc_ON_INIT
    hBitmap := BT_BitmapLoadFile ("HMG.bmp")
    BT_ClientAreaInvalidateAll ("Win1")
RETURN
```

```
PROCEDURE Proc_ON_RELEASE
   BT_BitmapRelease (hBitmap)
RETURN


PROCEDURE Proc_ON_PAINT
LOCAL  hDC, BTstruct
   hDC := BT_CreateDC ("Win1", BT_HDC_PAINTCLIENTAREA, @BTstruct)
        BT_DrawGradientFillVertical (hDC,;
                     0, 0,;
                     BT_ClientAreaWidth ("Win1"),;
                     BT_ClientAreaHeight("Win1"),;
                     WHITE, BLACK)
        BT_DrawBitmap (hDC, 35, 200, 300, 250, BT_COPY, hBitmap)
   BT_DeleteDC (BTstruct)
RETURN
```

6) *Bos Taurus:* Functions Reference Guide.

A) *Information about the Library*

BT_InfoName ()

Return a string with the name of the library: "Bos Taurus"

BT_InfoVersion ()

Return a string with the current version of the library, for example: "1.0.0"

BT_InfoAuthor ()

Return a string with the name of the author: "(c) Dr. Claudio Soto (from Uruguay)"

B) *Information about the Environment*

BT_GetDesktopHandle ()

Return a handle to the desktop (hWin). The desktop window covers the entire screen.

BT_DesktopWidth ()

Return the width of the screen in pixels.

BT_DesktopHeight ()

Return the height of the screen in pixels.

BT_WindowWidth *(Win)*

Return the width of the specified window in pixels.

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window.

BT_WindowHeight *(Win)*

Return the height of the specified window in pixels.

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window.

BT_ClientAreaWidth *(Win)*

Return the width of the client area of the specified window in pixels. The client area includes the area of the Status Bar.

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window.

BT_ClientAreaHeight *(Win)*

Return the height of the client area of the specified window in pixels. The client area includes the area of the Status Bar.

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window.

BT_StatusBarWidth *(cFormName)*

Return the width of the Status Bar of the specified window in pixels. If not defined (not exist) the Status Bar, returns zero.

cFormName: is the name of the window(e.g. "Win1").

BT_StatusBarHeight *(cFormName)*

Return the height of the Status Bar of the specified window in pixels. If not defined (not exist) the Status Bar, returns zero.

cFormName: is the name of the window(e.g. "Win1").

C) *Repaint the Client Area Functions*

BT_ClientAreaInvalidateAll *(Win, lErase)*

Force to redraw the all of the client area of the specified window.

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window.

lErase: specifies whether the background within the update region is to be erased when the update region is processed. If this parameter is TRUE (.T.), the background is

erased. If this parameter is FALSE (.F.), the background remains unchanged. For default: *lErase* = .F.

BT_ClientAreaInvalidateRect *(Win, Row, Col, Width, Height, lErase)*

Force to redraw the specified rectangle of the client area of the specified window.

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window.

Row, Col, Width, Height: specifies the dimensions in pixels of the rectangle to redraw. For default: *Row* = 0, *Col* = 0, *Width* = BT_ClientAreaWidth(Win), *Height* = BT_ClientAreaHeight(Win).

lErase: specifies whether the background within the update region is to be erased when the update region is processed. If this parameter is TRUE (.T.), the background is erased. If this parameter is FALSE (.F.), the background remains unchanged. For default: *lErase* = .F.

D) Bitmap Functions

BT_BitmapLoadFile *(cFileName)*

Loads an image (BMP, JPG, GIF, TIF or PNG) from the disk or resources and returns a handle to bitmap format image (*hBitmap*).

cFileName: is the name of the file or of the resource that contains the image.

BT_BitmapLoadEMF *(cFileName, aRGBcolor_Fill_Bk, NewWidht, NewHeight, Mode_Stretch)*

Loads an EMF (Enhanced Meta File) image from the disk or resources and returns a handle to bitmap format image (*hBitmap*).

cFileName: is the name of the file or of the resource that contains the image.

aRGBcolor_Fill_Bk: array containing the RGB colors that paint the background of the bitmap. For default: *aRGBcolor_Fill_Bk* = {0,0,0} = BLACK.

New_Width, New_Height: is the new size of the bitmap image. For default this values are the original width and height of the EMF image.

Mode_Stretch: sets the mode as the image of origin is adjusts (is stretches or compresses) in the new size bitmap, it is one of the constants: BT_SCALE or BT_STRETCH (defined in BosTaurus.CH). For default *Mode_Stretch* = BT_STRETCH.

BT_BitmapSaveFile *(hBitmap, cFileName, nTypePicture)*

Save an image (BMP, JPG, GIF, TIF or PNG) in the disk.

hBitmap: is a handle to the bitmap image.

cFileName: is the name of the file to save.

nTypePicture: specifies the format in which you want to save the image, it is one of the constants: BT_FILEFORMAT_BMP, BT_FILEFORMAT_JPG, BT_FILEFORMAT_GIF, BT_FILEFORMAT_TIF or BT_FILEFORMAT_PNG (defined in BosTaurus.CH). For default *nTypePicture* = BT_FILEFORMAT_BMP.

BT_BitmapRelease *(hBitmap)*

Release a bitmap of the memory.

hBitmap: is a handle to bitmap.

Note: all of the handles of bitmap obtained from load, create, clone, copy, etc. a bitmap image is responsibility of the application for his release before close.

BT_BitmapWidth *(hBitmap)*

Return the width of the specified bitmap in pixels.

hBitmap: is a handle to bitmap.

BT_BitmapHeight *(hBitmap)*

Return the height of the specified bitmap in pixels.

hBitmap: is a handle to bitmap.

BT_BitmapBitsPerPixel *(hBitmap)*

Return the bits per pixel of the specified bitmap.

hBitmap: is a handle to bitmap.

BT_BitmapCreateNew *(Width, Height, aRGBcolor_Fill_Bk)*

Creates a bitmap in the memory and returns a handle to bitmap (*hBitmap*).

Width, Height: are the dimensions of the bitmap in pixels.

aRGBcolor_Fill_Bk: array containing the RGB colors that paint the background of the bitmap. For default: *aRGBcolor_Fill_Bk* = {0,0,0} = BLACK.

BT_BitmapClone *(hBitmap, Row, Col, Width, Height)*

Clones the specified bitmap and returns a handle to the cloned bitmap.

hBitmap: is a handle to the origin bitmap.

Row, Col, Width, Height: specifies the dimensions in pixels of the rectangle to clone. For default: *Row* = 0, *Col* = 0, *Width* = BT_BitmapWidth(hBitmap), *Height* = BT_BitmapHeight(hBitmap).

BT_BitmapCopyAndResize *(hBitmap, New_Width, New_Height, Mode_Stretch, Algorithm)*

Copy and resizes the specified bitmap and returns a handle to the bitmap with the new size.

hBitmap: is a handle to the origin bitmap.

New_Width, New_Height: is the new size of the bitmap image.

Mode_Stretch: sets the mode as the bitmap of origin is adjusts (is stretches or compresses) in the new size bitmap, it is one of the constants: BT_SCALE or BT_STRETCH (defined in BosTaurus.CH). For default *Mode_Stretch* = BT_STRETCH.

Algorithm: sets the mode as the bitmap is resized, it is one of the constants: BT_RESIZE_COLORONCOLOR, BT_RESIZE_HALFTONE or BT_RESIZE_BILINEAR (defined in BosTaurus.CH). For default *Algorithm* = BT_RESIZE_HALFTONE.

BT_BitmapPaste *(hBitmap_D, Row_D, Col_D, Width_D, Height_D, Mode_Stretch, hBitmap_O)*

Paste a bitmap (origin) into another bitmap (destination).

hBitmap_D: is a handle to the destination bitmap.

Row_D, Col_D, Width_D, Height_D: specifies the size of the rectangle in pixels in the destination bitmap where you will paste the origin bitmap. For default: *Row_D* = 0, *Col_D* = 0, *Width* = BT_BitmapWidth(hBitmap_D), *Height* = BT_BitmapHeight(hBitmap_D).

Mode_Stretch: sets the mode as the bitmap of origin is adjusts (is stretches or compresses) in the specified rectangle in the destination bitmap, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hBitmap_O: is a handle to the origin bitmap.

BT_BitmapPasteTransparent *(hBitmap_D, Row_D, Col_D, Width_D, Height_D, Mode_Stretch, hBitmap_O, aRGBcolor_transp)*

Paste a transparent bitmap (origin) into another bitmap (destination).

hBitmap_D: is a handle to the destination bitmap.

Row_D, Col_D, Width_D, Height_D: specifies the size of the rectangle in pixels in the destination bitmap where you will paste the origin bitmap. For default: *Row_D* = 0, *Col_D* = 0, *Width* = BT_BitmapWidth(hBitmap_D), *Height* = BT_BitmapHeight(hBitmap_D).

Mode_Stretch: sets the mode as the bitmap of origin is adjusts (is stretches or compresses) in the specified rectangle in the destination bitmap, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hBitmap_O: is a handle to the origin bitmap.

aRGBcolor_transp: array containing the RGB colors to treat as transparent in the origin bitmap. For default: *aRGBcolor_transp* = color of the first pixel of the origin bitmap.

BT_BitmapPasteAlphaBlend *(hBitmap_D, Row_D, Col_D, Width_D, Height_D, Alpha, Mode_Stretch, hBitmap_O)*

Paste a bitmap (origin) with Alpha Blend effect (that make the pixels to be transparent or semi transparent) into another bitmap (destination).

hBitmap_D: is a handle to the destination bitmap.

Row_D, Col_D, Width_D, Height_D: specifies the size of the rectangle in pixels in the destination bitmap where you will paste the origin bitmap. For default: *Row_D* = 0, *Col_D* = 0, *Width* = BT_BitmapWidth(hBitmap_D), *Height* = BT_BitmapHeight(hBitmap_D).

Alpha: is alpha blending value to be applied to the entire source bitmap (range 0 to 255, transparent = 0, opaque = 255).

Mode_Stretch: sets the mode as the bitmap of origin is adjusts (is stretches or compresses) in the specified rectangle in the destination bitmap, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hBitmap_O: is a handle to the origin bitmap.

BT_BitmapInvert *(hBitmap)*

Invert the colors of an image (makes a negative of the original image).

hBitmap: is a handle to bitmap.

BT_BitmapGrayness *(hBitmap, Gray_Level)*

Set the grayness of the specified bitmap.

hBitmap: is a handle to bitmap.

Gray_Level: is the gray level in percentage (range 0 to 100%, none = 0, full = 100).

BT_BitmapBrightness *(hBitmap, Light_Level)*

Change the brightness of the specified bitmap.

hBitmap: is a handle to bitmap.

*Light_Level***:** is the light level (range -255 to +255, black = -255, white = +255, none = 0).

BT_BitmapContrast *(hBitmap, ContrastAngle)*

Increase the contrast of the colors in an image.

hBitmap: is a handle to bitmap.

ContrastAngle: is the angle of the slope (in radians) of the contrast transform.

BT_BitmapModifyColor *(hBitmap, RedLevel, GreenLevel, BlueLevel)*

Increases or decreases the colors channels of an image.

hBitmap: is a handle to bitmap.

RedLevel: is the Red color level to change (range -255 to +255, none = 0).

GreenLevel: is the Green color level to change (range -255 to +255, none = 0).

BlueLevel: is the Blue color level to change (range -255 to +255, none = 0).

BT_BitmapGammaCorrect *(hBitmap, RedGamma, GreenGamma, BlueGamma)*

Image displaying suffers from photometric distortions caused by the nonlinear response of display devices to lightness. The photometric response of a displaying device is known as the gamma response characteristic. Display monitors for different operating systems use different gammas. To compensate for these differences, the

gamma of the image needs to be corrected. A gamma correction with gamma equal to 1.0 is an identity color transformation. Gamma less than 1.0 makes a picture look darker, and gamma larger than 1.0 makes a picture look lighter.

hBitmap: is a handle to bitmap.

RedGamma: is the Gamma Red color level to change (normally range between 0.2 and 5.0, none = 1.0).

GreenGamma: is the Gamma Green color level to change (normally range between 0.2 and 5.0, none = 1.0).

 BlueGamma: is the Gamma Blue color level to change (normally range between 0.2 and 5.0, none = 1.0).

BT_BitmapConvolutionFilter3x3 *(hBitmap, aFilter)*

The convolution is a matrix (or two-dimensional arrangement) applied to an image. The elements of this array are integers values. The result of this operation is a new image that has been filtered. The convolution basically modifies the color of a pixel based on the color of the neighboring pixels. For each color channel, the color value for each pixel is calculated on the original color and the color of the surrounding pixels.

hBitmap: is a handle to bitmap.

aFilter: is an array with the following values:

{ k1, k2, k3,
  k4, k5, k6,
  k7, k8, k9,
  Divisor, Bias }

Where **k1 ... k9** are the values of the 3x3 matrix that multiply the pixels. The center cell value (k5) multiplies the pixel that is currently processing. **Divisor** determines the brightness of the final image and **Bias** increments (positive value) or decrements (negative value) the end color of the currently processed pixel. For more details see the demo10.

```
-------------------------------------------------------
     RULES OF THUMB TO CREATE USER DEFINED FILTERS
-------------------------------------------------------

        Center Cell Value      Surrounding Cell Values
             (k5)              (k1, k2, k3, k4, k6, k7, k8, k9)

        ----------------  ------------------------------------
Blur     POSITIVE         Symmetrical pattern of POSITIVE values

Sharpen  POSITIVE         Symmetrical pattern of NEGATIVE values

Edges    NEGATIVE         Symmetrical pattern of POSITIVE values

Emboss   POSITIVE         Symmetrical pattern of NEGATIVE values
                          on one side and POSITIVE values on the other

-------------------------------------------------------
```

Center Cell Value = k5

Surrounding Cell Values = k1, k2, k3, k4, k6, k7, k8, k9

Sum = (Center Cell Value) + (Surrounding Cell Values)

Sum = k1 + k2 + k3 + k4 + k5 + k6 + k7 + k8 + k9

If Sum / Divisor = 1 ---> Retain the brightness of the original image.

If Sum / Divisor > 1 ---> Increases the bright of the image

If Sum / Divisor < 1 ---> Darken the image

To reduce the effect of a filter (Blur, Sharpen, Edges, etc.) you must increase the value of the center cell (k5)

BT_BitmapTransform *(hBitmap, Mode, Angle, aRGBColor_Fill_Bk)*

Reflects and/or rotates of the specified bitmap and returns a handle to the new transformed bitmap.

hBitmap: is a handle to the origin bitmap.

Mode: is one or a combination (sum) of the constants BT_BITMAP_REFLECT_HORIZONTAL, BT_BITMAP_REFLECT_VERTICAL and BT_BITMAP_ROTATE (defined in BosTaurus.CH).

Angle: is the angle at which it is to rotate the image (range 0 to 360º). This parameter is considered only if BT_BITMAP_ROTATE is set in *Mode*, otherwise it is ignored.

aRGBColor_Fill_Bk: array containing the RGB colors to fill the empty spaces of the background of the image. This parameter is considered only if BT_BITMAP_ROTATE is

set in *Mode*, otherwise it is ignored. For default: *aRGBcolor_Fill_Bk* = color of the first pixel of the origin bitmap.

BT_BitmapCaptureDesktop *(Row, Col, Width, Height)*

Captures the desktop and returns a handle to the image bitmap captured (hBitmap).

Row, Col, Width, Height: specifies the dimensions in pixels of the rectangle to capture.

For default: *Row* = 0, *Col* = 0, *Width* = BT_DesktopWidth(), *Height* = BT_DesktopHeight().

BT_BitmapCaptureWindow *(Win, Row, Col, Width, Height)*

Captures a specified window and returns a handle to the image bitmap captured (hBitmap).

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window.

Row, Col, Width, Height: specifies the dimensions in pixels of the rectangle to capture.

For default: *Row* = 0, *Col* = 0, *Width* = BT_WindowWidth(Win), *Height* = BT_WindowHeight(Win).

BT_BitmapCaptureClientArea *(Win, Row, Col, Width, Height)*

Captures the client area of the specified window and returns a handle to the image bitmap captured (hBitmap).

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window.

Row, Col, Width, Height: specifies the dimensions in pixels of the rectangle to capture.

For default: *Row* = 0, *Col* = 0, *Width* = BT_ClientAreaWidth(Win), *Height* = BT_ClientAreaHeight(Win).

BT_BitmapClipboardGet *(Win)*

Gets a bitmap image from the clipboard and returns a handle to the bitmap (hBitmap). If the function fails or the clipboard is empty returns zero.

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window associated to opening of the clipboard.

BT_BitmapClipboardPut *(Win, hBitmap)*

Put a bitmap image in the clipboard. If the function fails returns FALSE (.F.), otherwise returns TRUE (.T.).

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window associated to opening of the clipboard.

hBitmap: is a handle to bitmap.

BT_BitmapClipboardClean *(Win)*

Empties the clipboard and frees all the handles of data in the clipboard (bitmap, text, metafiles, etc.). If the function fails or the clipboard is empty returns FALSE (.F.), otherwise returns TRUE (.T.).

Win: is the name (e.g. "Win1") or is the handle (e.g. hWin) of the window associated to opening of the clipboard.

BT_BitmapClipboardIsEmpty *()*

Returns TRUE (.T.) if exist a bitmap image in the clipboard, otherwise returns FALSE (.F.).

E) *Functions that Get/Release Handle Device Context*

BT_CreateDC *(Win_or_hBitmap, Type, @BTstruct)*

Returns the handle to the device context specified (hDC).

Type: specifies the type of the handle device context to get, can be one of the following constants: BT_HDC_DESKTOP (hDC to desktop), BT_HDC_WINDOW (hDC to all window), BT_HDC_ALLCLIENTAREA (hDC to the all client area), BT_HDC_INVALIDCLIENTAREA (hDC to the region of the client area that is not valid: invalid client area, i.e. needs to be repainted) or BT_HDC_BITMAP (hDC to a bitmap) (defined in BosTaurus.CH).

Win_or_hBitmap: is the name (e.g. "Win1") or is the handle (e.g. hWin) to the window if *Type* is set as BT_HDC_WINDOW, BT_HDC_ALLCLIENTAREA or BT_HDC_INVALIDCLIENTAREA. *Win_or_hBitmap* is a handle to bitmap (hBitmap) if *Type* is set as BT_HDC_BITMAP. If *Type* is set as BT_HDC_DESKTOP, *Win_or_hBitmap* is ignored.

BTstruct: is a variable (array) passed by reference that stores information about the hDC obtained. This information is necessary to properly release the hDC obtained.

BT_DeleteDC *(BTstruct)*

Release from the memory a handle of device context.

BTstruct: is an array with information about a determined handle of device context returned by **BT_CreateDC** function.

Remarks

The **Bos Taurus** graphic library uses the *common* Device Contexts (DC). The common DCs are *Display* device contexts maintained in a special cache by the system. Because only a limited number of common device contexts exist, an application should release them after it has finished drawing. Because new display device contexts for the cache are allocated in the application's heap space, failure to release the device contexts eventually consumes all available heap space this causes an error when it cannot allocate space for the new device context. **This means that you should get the hDC, draw and then immediately release the hDC, do not store the hDC to process other events drawing later.**

The best way to process the ON PAINT event in HMG is get the handle of the Device Context (hDC) of the client area with *Type* = BT_HDC_INVALIDCLIENTAREA in the BT_CreateDC function. This way is fast for drawing and causes less display flicker that with BT_HDC_ALLCLIENTAREA. The handle of the DC obtained with BT_HDC_ALLCLIENTAREA is indicated to paint the client area out of the ON PAINT event.

F) *Functions for Drawing on a Device Context*

BT_DrawGetPixel *(hDC, Row, Col)*

Gets RGB color value of the pixel at the specified coordinates to the Device Context (DC) specified and returns a array with the color = {R,G,B}.

hDC: is a handle to the device context.

Row, Col: specifies the coordinates in pixels in the DC.

BT_DrawSetPixel *(hDC, Row, Col, aRGBcolor)*

Sets RGB color value of the pixel at the specified coordinates to the Device Context (DC) specified.

hDC: is a handle to the device context.

Row, Col: specifies the coordinates in pixels in the DC.

aRGBcolor: is a array with the RGB color = {R,G,B}.

BT_DrawBitmap *(hDC, Row, Col, Width, Height, Mode_Stretch, hBitmap)*

Draws a bitmap in the Device Context (DC) specified.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the size of the rectangle in pixels in the DC where you will draw the bitmap. For default: *Row* = 0, *Col* = 0, *Width* = BT_BitmapWidth(hBitmap), *Height* = BT_BitmapHeight(hBitmap).

Mode_Stretch: sets the mode as the bitmap is adjusts (is stretches or compresses) in the specified rectangle in the DC, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hBitmap: is a handle to the bitmap.

BT_DrawBitmapTransparent *(hDC, Row, Col, Width, Height, Mode_Stretch, hBitmap, aRGBcolor_transp)*

Draws a transparent bitmap in the Device Context (DC) specified.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the size of the rectangle in pixels in the DC where you will draw the bitmap. For default: *Row* = 0, *Col* = 0, *Width* = BT_BitmapWidth(hBitmap), *Height* = BT_BitmapHeight(hBitmap).

Mode_Stretch: sets the mode as the bitmap is adjusts (is stretches or compresses) in the specified rectangle in the DC, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hBitmap: is a handle to the bitmap.

aRGBcolor_transp: array containing the RGB colors to treat as transparent in the bitmap. For default: *aRGBcolor_transp* = color of the first pixel of the origin bitmap.

BT_DrawBitmapAlphaBlend *(hDC, Row, Col, Width, Height, Alpha, Mode_Stretch, hBitmap)*

Draw a bitmap with Alpha Blend effect (that make the pixels to be transparent or semi transparent) in the Device Context (DC) specified.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the size of the rectangle in pixels in the DC where you will draw the bitmap. For default: *Row* = 0, *Col* = 0, *Width* = BT_BitmapWidth(hBitmap), *Height* = BT_BitmapHeight(hBitmap).

Alpha: is alpha blending value to be applied to the entire source bitmap (range 0 to 255, transparent = 0, opaque = 255).

Mode_Stretch: sets the mode as the bitmap is adjusts (is stretches or compresses) in the specified rectangle in the DC, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hBitmap: is a handle to the bitmap.

BT_DrawDCtoDC *(hDC1, Row1, Col1, Width1, Height1, Mode_Stretch, hDC2, Row2, Col2, Width2, Height2)*

Copy a rectangular area of a device context (origin) in another device context (destination).

hDC1: is a handle to the destination device context.

Row1, Col1, Width1, Height1: specifies the size of the rectangle in pixels in the destination DC where you will copy the origin DC.

Mode_Stretch: sets the mode as the rectangle in the origin DC is adjusts (is stretches or compresses) in the specified rectangle in the destination DC, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hDC2: is a handle to the origin device context.

Row2, Col2, Width2, Height2: specifies the size of the rectangle in pixels in the origin DC to copied in the destination DC.

BT_DrawDCtoDCTransparent *(hDC1, Row1, Col1, Width1, Height1, Mode_Stretch, hDC2, Row2, Col2, Width2, Height2, aRGBcolor_transp)*

Copy a transparent rectangular area of a device context (origin) in another device context (destination).

hDC1: is a handle to the destination device context.

Row1, Col1, Width1, Height1: specifies the size of the rectangle in pixels in the destination DC where you will copy the origin DC.

Mode_Stretch: sets the mode as the rectangle in the origin DC is adjusts (is stretches or compresses) in the specified rectangle in the destination DC, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hDC2: is a handle to the origin device context.

Row2, Col2, Width2, Height2: specifies the size of the rectangle in pixels in the origin DC to copied in the destination DC.

aRGBcolor_transp: array containing the RGB colors to treat as transparent in the origin DC. For default: *aRGBcolor_transp* = color of the first pixel of the origin DC.

BT_DrawDCtoDCAlphaBlend *(hDC1, Row1, Col1, Width1, Height1, Alpha, Mode_Stretch, hDC2, Row2, Col2, Width2, Height2)*

Copy a rectangular area of a device context (origin) with Alpha Blend effect (that make the pixels to be transparent or semi transparent) in another device context (destination).

hDC1: is a handle to the destination device context.

Row1, Col1, Width1, Height1: specifies the size of the rectangle in pixels in the destination DC where you will copy the origin DC.

Alpha: is alpha blending value to be applied to the entire source bitmap (range 0 to 255, transparent = 0, opaque = 255).

Mode_Stretch: sets the mode as the rectangle in the origin DC is adjusts (is stretches or compresses) in the specified rectangle in the destination DC, it is one of the constants: BT_SCALE, BT_STRETCH or BT_COPY (defined in BosTaurus.CH).

hDC2: is a handle to the origin device context.

Row2, Col2, Width2, Height2: specifies the size of the rectangle in pixels in the origin DC to copied in the destination DC.

BT_DrawGradientFillHorizontal *(hDC, Row, Col, Width, Height, aColorRGBstart, aColorRGBend)*

Fill a rectangular region from left to right (horizontal) in a Device Context (DC) with a background color that is interpolated from two colors specified.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the size of the rectangle in pixels in the DC where it will be filled.

aColorRGBstart: array that contains the RGB colors with that will begin filling. For default: *aColorRGBstart* = {0,0,0} = BLACK

aColorRGBend: array that contains the RGB colors with that will end filling. For default: *aColorRGBend* = {255,255,255} = WHITE

BT_DrawGradientFillVertical *(hDC, Row, Col, Width, Height, aColorRGBstart, aColorRGBend)*

Fill a rectangular region from top to bottom (vertical) in a Device Context (DC) with a background color that is interpolated from two colors specified.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the size of the rectangle in pixels in the DC where it will be filled.

aColorRGBstart: array that contains the RGB colors with that will begin filling. For default: *aColorRGBstart* = {255,255,255} = WHITE

aColorRGBend: array that contains the RGB colors with that will end filling. For default: *aColorRGBend* = {0,0,0} = BLACK

BT_DrawText *(hDC, Row, Col, cText, cFontName, nFontSize, aFontColor, aBackColor, nTypeText, nAlingText, nOrientation)*

Write a character string in the Device Context (DC) specified.

hDC: is a handle to the device context.

Row, Col: specifies the coordinates in pixels in the DC.

cText: is the text to draw.

cFontName: is the name of the font, e.g. "Times New Roman"

nFontSize: is the size of the font in logical units, e.g. 12.

aFontColor: array that contains the RGB colors that set the text color. For default *aFontColor* = {0,0,0} = BLACK

aBackColor: array that contains the RGB colors that set the background text color. For default *aBackColor* = {255,255,255} = WHITE

nTypeText: sets if background fills (opaque) or not (transparent) with the BackColor color before the text it draws, it is one of the constants: BT_TEXT_OPAQUE or BT_TEXT_TRANSPARENT (defined in BosTaurus.CH). For default: *nTypeText* = BT_TEXT_OPAQUE.

nAlingText: sets the horizontal and vertical alignment of the text with respect of specified coordinates in *Col* and *Row* respectively. The horizontal alignment is set with

one of the following constants: BT_TEXT_LEFT, BT_TEXT_CENTER or BT_TEXT_RIGHT (defined in BosTaurus.CH). The vertical alignment is set with one of the following constants: BT_TEXT_TOP, BT_TEXT_BASELINE or BT_TEXT_BOTTOM (defined in BosTaurus.CH). For default *nAlingText* = BT_TEXT_LEFT + BT_TEXT_TOP

nOrientation: sets the orientation of the text with respect of specified coordinates in *Col* and *Row* (range -90 to +90º). In BosTaurus.CH is defines some constants: BT_TEXT_NORMAL_ORIENTATION, BT_TEXT_VERTICAL_ASCENDANT, BT_TEXT_VERTICAL_DESCENDANT, BT_TEXT_DIAGONAL_ASCENDANT, BT_TEXT_DIAGONAL_DESCENDANT. For default *nOrientation* = BT_TEXT_NORMAL_ORIENTATION.

BT_DrawPolyLine *(hDC, aPointY, aPointX, aColorRGBLine, nWidthLine)*

Draws a series of line segments by connecting the points the arrays specified.

hDC: is a handle to the device context.

aPointY: array containing the coordinates in pixels of the points on the y-axis.

aPointX: array containing the coordinates in pixels of the points on the x-axis.

aColorRGBLine: array that contains the RGB colors that set of the lines color.

nWidthLine: width in pixels of the lines. For default: *nWidthLine* = 1.

BT_DrawLine *(hDC, Row1, Col1, Row2, Col2, aColorRGBLine, nWidthLine)*

Draw a line by connecting two points specified.

hDC: is a handle to the device context.

Row1, Col1: are the coordinates in pixels of the starting point.

Row2, Col2: are the coordinates in pixels of the end point.

aColorRGBLine: array that contains the RGB colors that set of the line color.

nWidthLine: width in pixels of the line. For default: *nWidthLine* = 1.

BT_DrawRectangle *(hDC, Row, Col, Width, Height, aColorRGBLine, nWidthLine)*

Draw a rectangle with the specified dimensions.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the dimensions in pixels of the rectangle to be draw.

aColorRGBLine: array that contains the RGB colors that set of the lines color.

nWidthLine: width in pixels of the lines. For default: *nWidthLine* = 1.

BT_DrawEllipse *(hDC, Row1, Col1, Width, Height, aColorRGBLine, nWidthLine)*

Draw an ellipse with the specified dimensions.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the dimensions in pixels of the ellipse to be draw.

aColorRGBLine: array that contains the RGB colors that set of the line color.

nWidthLine: width in pixels of the line. For default: *nWidthLine* = 1.

BT_DrawFillRectangle *(hDC, Row, Col, Width, Height, aColorRGBFill, aColorRGBLine, nWidthLine)*

Draw a fill rectangle with the specified dimensions.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the dimensions in pixels of the rectangle to be draw.

aColorRGBFill: array that contains the RGB colors that set of the fill color.

aColorRGBLine: array that contains the RGB colors that set of the lines color. For default: *aColorRGBLine = aColorRGBFill.*

nWidthLine: width in pixels of the lines. For default: *nWidthLine* = 1.

BT_DrawFillEllipse *(hDC, Row, Col, Width, Height, aColorRGBFill, aColorRGBLine, nWidthLine)*

Draw a fill ellipse with the specified dimensions.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the dimensions in pixels of the ellipse to be draw.

aColorRGBFill: array that contains the RGB colors that set of the fill color.

aColorRGBLine: array that contains the RGB colors that set of the line color. For default: *aColorRGBLine = aColorRGBFill.*

nWidthLine: width in pixels of the line. For default: *nWidthLine* = 1.

BT_DrawFillRoundRect *(hDC, Row, Col, Width, Height, RoundWidth, RoundHeight, aColorRGBFill, aColorRGBLine, nWidthLine)*

Draw a fill rectangle with rounded corners.

hDC: is a handle to the device context.

Row, Col, Width, Height: specifies the dimensions in pixels of the rectangle to be draw.

RoundWidth: specifies the width in pixels of the ellipse used to draw the rounded corners.

RoundHeight: specifies the height in pixels of the ellipse used to draw the rounded corners.

aColorRGBFill: array that contains the RGB colors that set of the fill color.

aColorRGBLine: array that contains the RGB colors that set of the line color. For default: *aColorRGBLine = aColorRGBFill.*

nWidthLine: width in pixels of the line. For default: *nWidthLine* = 1.

BT_DrawFillFlood *(hDC, Row, Col, aColorRGBFill)*

Fill an area in all directions defined by the color of the point of the specified coordinates.

hDC: is a handle to the device context.

Row, Col: specifies the coordinates in pixels of the point where begin filling.

aColorRGBFill: array that contains the RGB colors that set of the fill color.

G) Functions of Connection Between HMG Controls and Bos Taurus

BT_HMGGetImage *(cFormName, cControlName)*

Returns the handle of the bitmap associated to an Image Control of HMG (@...IMAGE).

cFormName: is the name (e.g. "Win1") of the parent window.

cControlName: is the name (e.g. "Image1") of the image control.

BT_ HMGCloneImage *(cFormName, cControlName)*

Clones the bitmap associated to an Image Control of HMG (@...IMAGE) and returns a handle to the cloned bitmap.

cFormName: is the name (e.g. "Win1") of the parent window.

cControlName: is the name (e.g. "Image1") of the image control.

BT_HMGSetImage *(cFormName, cControlName, hBitmap, lReleasePrevious)*

Sets a specified bitmap into an Image Control of HMG (@...IMAGE) and automatically releases the handle of the bitmap previously associated to the Image Control.

cFormName: is the name (e.g. "Win1") of the parent window.

cControlName: is the name (e.g. "Image1") of the image control.

hBitmap: is a handle to the bitmap to set.

lReleasePrevious: releases of the hBitmap previous associate to Image Control. For default is .T.

7) *Bos Taurus: List of the Functions*

BT_InfoName

BT_InfoVersion

BT_InfoAuthor

BT_GetDesktopHandle

BT_DesktopWidth

BT_DesktopHeight

BT_WindowWidth

BT_WindowHeight

BT_ClientAreaWidth

BT_ClientAreaHeight

BT_StatusBarWidth

BT_StatusBarHeight

BT_ClientAreaInvalidateAll

BT_ClientAreaInvalidateRect

BT_BitmapLoadFile

BT_BitmapLoadEMF

BT_BitmapSaveFile

BT_BitmapRelease

BT_BitmapWidth

BT_BitmapHeight

BT_BitmapBitsPerPixel

BT_BitmapCreateNew

BT_BitmapClone

BT_BitmapCopyAndResize

BT_BitmapPaste

BT_BitmapPasteTransparent

BT_BitmapPasteAlphaBlend

BT_BitmapInvert

BT_BitmapGrayness

BT_BitmapBrightness

BT_BitmapContrast

BT_BitmapModifyColor

BT_BitmapGammaCorrect

BT_BitmapConvolutionFilter3x3

BT_BitmapTransform

BT_BitmapCaptureDesktop

BT_BitmapCaptureWindow

BT_BitmapCaptureClientArea

BT_BitmapClipboardGet

BT_BitmapClipboardPut

BT_BitmapClipboardClean

BT_BitmapClipboardIsEmpty

BT_CreateDC

BT_DeleteDC

BT_DrawGetPixel

BT_DrawSetPixel

BT_DrawBitmap

BT_DrawBitmapTransparent

BT_DrawBitmapAlphaBlend

BT_DrawDCtoDC

BT_DrawDCtoDCTransparent

BT_DrawDCtoDCAlphaBlend

BT_DrawGradientFillHorizontal

BT_DrawGradientFillVertical

BT_DrawText

BT_DrawPolyLine

BT_DrawLine

BT_DrawRectangle

BT_DrawEllipse

BT_DrawFillRectangle

BT_DrawFillEllipse   **BT_HMGGetImage**

BT_DrawFillRoundRect   **BT_HMGCloneImage**

BT_DrawFillFlood   **BT_HMGSetImage**