Status Bar Control
Version 1.2.0

Jean-Marc Krikorian
October 8, 1991

Changes to this version of the Status Bar Control DLL:

The functions **StatusBarInit()** and **StatusBarUnInit()** no longer exist. Calls to the Windows SDK functions **LoadLibrary()** and **FreeLibrary()** replace these functions respectively. *(See "Accessing the Status Bar Control" below.)*

The default Status Bar styles are determined when the Status Bar is created. If the monitor is a 16 color VGA or better then the default Face Style is 3D and the default Face Color is gray. If, on the otherhand, the monitor is not 16 color VGA then the default Face Style is Flat and the default Face Color is white.

Also, the file STSBAR.LIB is no longer needed and is therefore not included.

Finally, the drawing routines have been optimized so that the flickering that occurred in previous versions is no longer existent.

The Status Bar Control DLL is a shareware product. I welcome any suggestions, comments or questions you may have on this product. The source code is also available. Please direct all requests to the address or telephone number below.

Jean-Marc Krikorian
525 Sandy Lane
Libertyville, IL 60048
(708) 816-3314

The Status Bar Control consists of two files:

1.) STSBAR.EXT. This is the external header file that contains the message IDs.
2.) STSBAR.DLL. This is the Status Bar DLL. Make sure it resides in your current directory or in a directory pointed to by your PATH environmental variable.

A sample application SBTEST.* is included along with its source. Refer to it when building your application. The sample application comes with a makefile to show how to link with the Status Bar DLL.

Introduction:

The Status Bar control defines a new window class. This window class as the name implies creates a status bar. The status bar is used to display messages to the user and/or to display current information. For an example refer to Microsoft Word for Windows. As long as the user does not interact with the menu bar the status bar displays such information as the page number and section number. However, when the user interacts with the menu bar the status bar displays more verbose information specific to each menu option.

Technical Issues:

The Status Bar by virtue of it being a control is implemented such that it is re-entrant. This means that the programmer can create multiple Status Bar controls within one application and that more than one application can have a Status Bar control. Think of the Status Bar control as a pushbutton. A dialog box can have many pushbuttons within it yet each pushbutton is unique in the sense that different actions can occur by selecting different pushbuttons. The way to differentiate between the different controls is through either their unique IDs or unique handles. To get a Status Bar the user must perform a CreateWindow() call using the class name "*StatusBarClass*" and a unique ID. Each CreateWindow() call will return a unique handle for that particular Status Bar control. Hence, by storing these handles the user can access multiple Status Bars.

The Status Bar attributes of Face Style, Face Color and Text Color will only work on a monitor that is 16 color VGA or better. The reason is that on monitors of lesser resolution and color capabilities the dithering causes the Status Bar Face Styles to be rather unattractive. Therefore, for these monitors, the Status Bar exhibits a white flat face and black text.

Status Bar Features:

With the Status Bar control you are able to:

1.) Choose a location for the control (Top, Bottom or Moveable)
2.) Pick the Face Style (3D, 2D or Flat)
3.) Pick the Face Color (Any RGB value)
4.) Pick the Font Family and Point Size
5.) Pick the Text Color (Any RGB value)
6.) Pick the Text Attributes (Bold, Italic, Underline and/or StrikeOut)
7.) Receive mouse click notification messages

Accessing the Status Bar Control:

To access the Status Bar control the following steps have to be taken:

1.) Include the header file "STSBAR.EXT". This header file contains the various messages and message parameters that are available.

2.) Before creating the window class make the following call:

hLib = LoadLibrary("STSBAR.DLL");

This call will take care of registering the Status Bar class. Also, save the handle (*hLib*) since it will be used to unload the library when you are done using the Status Bar. *(See step 5 below.)*

3.) Call the Windows function CreateWindow() and use the class name "*StatusBarClass*". Also, make sure to pass in a unique ID number, since this will be used to access notification messages that are passed back. The Status Bar window can be created with one of three styles:

a.) the SBS_BOTTOM style creates a Status Bar that will always be at the bottom of the parent's client area.

b.) the SBS_TOP style creates a Status Bar that will always be at the top of the parent's client area.

c.) the SBS_MOVEABLE style creates a Status Bar which can have a user chosen y-origin. (*see related information in the SBM_SETFONT and SBM_SIZE messages*).

4.) In the WM_SIZE message send the Status Bar an SBM_SIZE message and pass in the parent's size (which is contained in the lParam) as the lParam to the SendMessage() call. (i.e. SendMessage(hStatusBar, SBM_SIZE, 0, lParam)). (*For a Status Bar created with the SBS_MOVEABLE style, the wParam should contain the Status Bar's y-origin.*)

5.) When you are done using the Status Bar make the following call:

FreeLibrary(hLib);

This call will remove the DLL from memory and perform any necessary cleanup, such as unregistering the Status Bar class.

Status Bar Messages

SBM_DISPLAYTEXT

This message is used to display text inside the Status Bar.

Parameter
Description

wParam
Contains text attributes that can be 'Or'ed together. The values are
1.) SBTA_BOLD
2.) SBTA_ITALICS
3.) SBTA_UNDERLINE
4.) SBTA_STRIKEOUT

lParam
Contains a LPSTR to the text to be displayed.

Example
SendMessage(hStatusBar, SBM_DISPLAYTEXT, (WORD)(SBTA_BOLD | SBTA_UNDERLINE), (LONG)(LPSTR)"This is some text.");

Comments
The font used to display the text will be the current system font in use by Windows or the font passed in with the SBM_SETFONT message.
The Status Bar control will store its own copy of the text, so that the user does not have to bother with repaints to the Status Bar.
The default is to display the text without any of these attributes.
The text passed in can be any length, however, it will be clipped within the client area of the Status Bar.

Return Value
None.

Status Bar Messages continued...

SBM_FACECOLOR

This message is used to change the color of the display face of the Status Bar.

Parameter
Description

wParam
Not used.

lParam
Can contain either one of the Status Bar color defines in the header file, currently these are:
1.) SBCLR_RED
2.) SBCLR_GREEN
3.) SBCLR_WHITE
4.) SBCLR_BLACK
5.) SBCLR_GRAY

Yet, if you look at the definitions for these constants you will see that they are actually instances of the Windows macro **RGB(r,g,b)**. As a result, the lParam can contain this RGB() macro with any values in it and these values will be used for the color of the display face.

Example
SendMessage(hStatusBar, SBM_FACECOLOR, 0, (LONG)SBCLR_BLUE);
or
SendMessage(hStatusBar, SBM_FACECOLOR, 0, (LONG)RGB(50, 175, 210));

Comments
On monitors that are 16 color VGA or better, the default face color is gray. Otherwise, the default face color is white.

Return Value
Returns the previous display face color.

Status Bar Messages continued...

SBM_FACESTYLE

This message is used to change the face style of the Status Bar.

Parameter
Description

wParam
Not used.

lParam
Contains one value for the face style. These styles should not be 'Or'ed together.
1.) SBFS_3DFACE
2.) SBFS_2DFACE
3.) SBFS_FLATFACE

Example
SendMessage(hStatusBar, SBM_FACESTYLE, 0, SBFS_3DFACE);

Comments

On mon9
itors that are 16 color VGA or better the default is SBFS_3DFACE. Otherwise, the default is SBFS_FLATFACE.

Return Value
Returns the previous face style.

Status Bar Messages continued...

SBM_SETFONT

This message is sent to the Status Bar to tell it to display a particular font and point size.

Parameter
Description

wParam
Only used with a Status Bar created with the SBS_MOVEABLE style. If the control is defined with such a style then the wParam should contain the y-origin.

lParam
Contains the address to a LOGFONT structure that contains the font information. When the Status Bar is first created and if no particular font is selected, then the Status Bar selects the default system font to display text.
To reset the Status Bar so that it uses the default system font set lParam to zero.

Example
SendMessage(hStatusBar, SBM_SETFONT, 0, (LONG)(LPLOGFONT)&lf);

If the Status Bar is created with the SBS_MOVEABLE style then use:
SendMessage(hStatusBar, SBM_SETFONT, (WORD)yorigin, (LONG)(LPLOGFONT)&lf);

To reset the Status Bar so that it uses the default system font the following is done:
SendMessage(hStatusBar, SBM_SETFONT, 0, 0L);

Comments
Only two fields in the log font structure are used these are:
1.) lf.lfHeight
2.) lf.lfFaceName

Return Value
None.

Status Bar Messages continued...

SBM_SIZE

This message is sent to the Status Bar to tell it to resize itself. This message should be sent from inside the WM_SIZE message of the parent, since the Status Bar sizes itself with respect to the width of the parent.

Parameter
Description

wParam
Only used with a Status Bar created with the SBS_MOVEABLE style. If the control is defined with such a style then the wParam should contain the y-origin.

lParam
Contains the new width and height of the client area of the window. Refer to WM_SIZE in the SDK manual.
If this message is called from within the parent's WM_SIZE, pass in the lParam. Since it already contains the parent's size. Otherwise you must determine the size of the parent and pass that value in.

Example
SendMessage(hStatusBar, SBM_SIZE, 0, (LONG)lSize);

If the Status Bar is created with the SBS_MOVEABLE style then use:
SendMessage(hStatusBar, SBM_SIZE, (WORD)yorigin, (LONG)lSize);

Comments
None.

Return Value
None.

Status Bar Messages continued...

SBM_TEXTCOLOR

This message is used to change the color of the display text inside the Status Bar.

Parameter
Description

wParam
Not used.

lParam
Can contain either one of the Status Bar color defines in the header file, currently these are:
1.) SBCLR_RED
2.) SBCLR_GREEN
3.) SBCLR_WHITE
4.) SBCLR_BLACK
5.) SBCLR_GRAY

Yet, if you look at the definitions for these constants you will see that they are actually instances of the Windows macro **RGB(r,g,b)**. As a result, the lParam can contain this RGB() macro with any values in it and these values will be used to display the text color.

Example
SendMessage(hStatusBar, SBM_TEXTCOLOR, 0, (LONG)SBCLR_BLUE);
or
SendMessage(hStatusBar, SBM_TEXTCOLOR, 0, (LONG)RGB(50, 175, 210));

Comments
The default text color is black.

Return Value
Returns the previous text color.

Status Bar Notification Messages

SBN_LBUTTONDBLCLK

This code specifies that the Status Bar control received a left button double-click. The parent receives this code through the WM_COMMAND message from the control.

Parameter
Description

wParam
Specifies the control ID of the Status Bar.

lParam
Contains the SBN_LBUTTONDBLCLK code in the high-order word.

Example

```
...
case WM_COMMAND:
{
        switch(wParam)
        {
                case STATUSBAR_ID:
                        switch(HIWORD(lParam))
                        {
                                case SBN_LBUTTONDBLCLK:
                                        /* processing occurs here */
                                        break;
                        }
        }
}
...
```

Comments
None.

Return Value
None.

Status Bar Notification Messages continued...

SBN_LBUTTONUP

This code specifies that the Status Bar control received a left button up message. The parent receives this code through the WM_COMMAND message from the control.

Parameter
Description

wParam
Specifies the control ID of the Status Bar.

lParam
Contains the SBN_LBUTTONUP code in the high-order word.

Example

```
...
case WM_COMMAND:
{
        switch(wParam)
        {
                case STATUSBAR_ID:
                        switch(HIWORD(lParam))
                        {
                                case SBN_LBUTTONUP:
                                        /* processing occurs here */
                                        break;
                        }
        }
}
...
```

Comments
None.

Return Value
None.

Status Bar Notification Messages continued...

SBN_MBUTTONDBLCLK

This code specifies that the Status Bar control received a middle button double-click. The parent receives this code through the WM_COMMAND message from the control.

Parameter
Description

wParam
Specifies the control ID of the Status Bar.

lParam
Contains the SBN_MBUTTONDBLCLK code in the high-order word.

Example

```
...
case WM_COMMAND:
{
        switch(wParam)
        {
                case STATUSBAR_ID:
                        switch(HIWORD(lParam))
                        {
                                case SBN_MBUTTONDBLCLK:
                                        /* processing occurs here */
                                        break;
                        }
        }
}
...
```

Comments
None.

Return Value
None.

Status Bar Notification Messages continued...

SBN_MBUTTONUP

This code specifies that the Status Bar control received a middle button up message. The parent receives this code through the WM_COMMAND message from the control.

Parameter
Description

wParam
Specifies the control ID of the Status Bar.

lParam
Contains the SBN_MBUTTONUP code in the high-order word.

Example

```
...
case WM_COMMAND:
{
        switch(wParam)
        {
                case STATUSBAR_ID:
                        switch(HIWORD(lParam))
                        {
                                case SBN_MBUTTONUP:
                                        /* processing occurs here */
                                        break;
                        }
        }
}
...
```

Comments
None.

Return Value
None.

Status Bar Notification Messages continued...

SBN_RBUTTONDBLCLK

This code specifies that the Status Bar control received a right button double-click. The parent receives this code through the WM_COMMAND message from the control.

Parameter
Description

wParam
Specifies the control ID of the Status Bar.

lParam
Contains the SBN_RBUTTONDBLCLK code in the high-order word.

Example

```
...
case WM_COMMAND:
{
        switch(wParam)
        {
                case STATUSBAR_ID:
                        switch(HIWORD(lParam))
                        {
                                case SBN_RBUTTONDBLCLK:
                                        /* processing occurs here */
                                        break;
                        }
        }
}
...
```

Comments
None.

Return Value
None.

Status Bar Notification Messages continued...

SBN_RBUTTONUP

This code specifies that the Status Bar control received a right button up message. The parent receives this code through the WM_COMMAND message from the control.

Parameter
Description

wParam
Specifies the control ID of the Status Bar.

lParam
Contains the SBN_RBUTTONUP code in the high-order word.

Example

```
...
case WM_COMMAND:
{
        switch(wParam)
        {
                case STATUSBAR_ID:
                        switch(HIWORD(lParam))
                        {
                                case SBN_RBUTTONUP:
                                        /* processing occurs here */
                                        break;
                        }
        }
}
...
```

Comments
None.

Return Value
None.

Status Bar Example Code

```
#include "stsbar.ext"

#define ID_STATUSBAR 2000

HANDLE hLib;

int PASCAL WinMain(...)
{
.
.
.
hWndParent = CreateWindow(...);

// Load the Status Bar Control DLL
hLib = LoadLibrary("STSBAR.DLL");

// Create the Status Bar control
hStatusBar = CreateWindow(      "StatusBarClass",
                                NULL,
                                WS_CHILD | SBS_BOTTOM,
                                0,0,0,0,
                                hWnd,
                                ID_STATUSBAR,
                                hInstance,
                                NULL);

// The default styles for the Status Bar control depend on the monitor type and are:
// If the monitor is 16 color VGA or better then: 3D Face Style and Gray Face Color.
// Otherwise: Flat Face Style and White Face Color.
// The default Text Color is black.

// To change any of these styles before showing the Status Bar
// use SendMessage() calls to change any or all of these attributes
// and make sure not to use WS_VISIBLE when creating the Status Bar

                        // For example, the following calls change the Status Bar
to have a
// gray 3D face with black text color
SendMessage(hStatusBar, SBM_FACESTYLE, 0, (LONG)SBFS_3DFACE);
SendMessage(hStatusBar, SBM_FACECOLOR, 0, (LONG)SBCLR_GRAY);
SendMessage(hStatusBar, SBM_TEXTCOLOR, 0, (LONG)SBCLR_BLACK);

// Now show the Status Bar
ShowWindow(hStatusBar, SW_SHOW);
.
.
.
}
```

```
LONG FAR PASCAL ParentWndProc(...)
{
        switch (message)
        {
                case WM_COMMAND:
                        switch (wParam)
                        {
                                // How to change the face style in the Status Bar
                                case IDM_FACESTYLE:
                                        SendMessage(hStatusBar, SBM_FACESTYLE, 0,
                                                        (LONG)SBFS_FLATFACE);
                                        return 0;

                                // How to change the face color in the Status Bar
                                case IDM_FACECOLOR:
                                        SendMessage(hStatusBar, SBM_FACECOLOR, 0,
                                                        (LONG)SBCLR_BLUE);
                                        or
                                        SendMessage(hStatusBar, SBM_FACECOLOR, 0,
                                                        (LONG)RGB(0, 0, 255));
                                        return 0;

                                // How to change the text color in the Status Bar
                                case IDM_TEXTCOLOR:
                                        SendMessage(hStatusBar, SBM_TEXTCOLOR, 0,
                                                        (LONG)SBCLR_RED);
                                        or
                                        SendMessage(hStatusBar, SBM_TEXTCOLOR, 0,
                                                        (LONG)RGB(255, 0, 0));
                                        return 0;

                                // How to change the text font in the Status Bar
                                case IDM_HELV8:
                                {
                                        LOGFONT lf;

                                        lf.lfHeight = 8;
                                        lf.lfWidth = 0;                 // this field is ignored
                                        lf.lfEscapement = 0;            // this field is ignored
                                        lf.lfOrientation = 0;           // this field is ignored
                                        lf.lfWeight = 0;                // this field is ignored
                                        lf.lfItalic = 0;                // this field is ignored
                                        lf.lfUnderline = 0;             // this field is ignored
                                        lf.lfStrikeOut = 0;             // this field is ignored
                                        lf.lfCharSet = 0;               // this field is ignored
                                        lf.lfOutPrecision = 0;          // this field is ignored
                                        lf.lfClipPrecision = 0;         // this field is ignored
                                        lf.lfQuality = 0;               // this field is ignored
                                        lf.lfPitchAndFamily = 0;        // this field is ignored
                                        lstrcpy(lf.lfFaceName, "Helv");
                                        SendMessage(hStatusBar, SBM_SETFONT, 0,
                                                        (LONG)(LOGFONT)&lf);
                                        return 0;
                                }
```

```
                                        // How to reset the font in the Status Bar to the default system font
                                        case IDM_DEFAULTSYSTEM:
                                                SendMessage(hStatusBar, SBM_SETFONT, 0, 0L);
                                                return 0;

                                        // How to display text in the Status Bar
                                        case IDM_DISPLAYTEXT:
                                                SendMessage(hStatusBar, SBM_DISPLAYTEXT,
                                                        (WORD)(SBTA_ITALICS |
                                                                SBTA_UNDERLINE),
                                                        (LONG)(LPSTR)lpszText);
                                                return 0;

                                        // How to intercept mouse messages in the Status Bar
                                        case ID_STATUSBAR:
                                                switch (HIWORD(lParam))
                                                {
                                                        // How to intercept a left mouse button up in the
                                                        // Status Bar
                                                        case SBN_LBUTTONUP:
                                                        return 0;

                                                        // How to intercept a left mouse button double click
                                                        // in the Status Bar
                                                        case SBN_LBUTTONDBLCLK:
                                                        return 0;
                                                }
                                                return 0;
                                }
                                return 0;

                        // How to size the Status Bar
                        case WM_SIZE:
                                if (hStatusBar)
                                        SendMessage(hStatusBar, SBM_SIZE, 0, lParam);
                                return 0;

                        // How to Remove the DLL from memory
                        case WM_DESTROY:
                                FreeLibrary(hLib);
                                PostQuitMessage(0);
                                return 0;
        }

return (DefWindowProc(...));
}
```