Tool Bar /Status Bar control libraries (v 2.01).

0. For a quick start run DEMOTOOL.EXE or read the following information.

Help pages for individual functions and messages were generated by Autodoc.

About.

ESTOOLS.DLL is a dynamic link library which allows a programmer to create ToolBar and StatusBar controls (ToolBar from resource script or run time, StatusBar run time only) and incorporate them into any MSWin 3.x application. It takes no more programming than usual dialog boxes or menus. The Status Bar is a control (usually on the bottom of the parent window) which lets you output text in the same manner as printf() functions. The library is shipped along with several files:

readme.wri -- this file;
estools.dll -- Tool Bar/Status bar DLL
estools.lib -- Import library.
demotool.exe -- Demo program which uses estools.dll and shows its abilities.
demotool.ide -- BC++ 4.0 project file to build demotool.exe -- BE CAREFUL, YOUR DIRECTORIES WILL BE DIFFERENT ! ! !
makefile Make file to build a demotool with BC++.
demofunc.obj -- This module contains calls to SendMessage function with messages undocumented in demo version.
demofunc.c -- Source for demofunc.obj. If you are using anything else than BC, you better use it to create your own demofunc.obj.
demotool.c -- C source for demotool.exe;
esdefs.h -- C header with various definitions specific for demotool.c;
estools.h -- Header file containing definitions specific for estools.dll- it contains most of the information on how to use the DLL;
estools.def -- Module definition file. You do not really need it but you may want to use it to make your own import lib;
demotool.rc -- Resource script file;
bmp0.bmp, bmp2_5.bmp,
bmp6.bmp,
btns.bmp-- Pictures for button's tops.
tbdemo.ico -- Application Icon;
invoice.txt -- Some kind of invoice. Ignore it if you have paid already.

All detailed information required to use Tool Bar and Status Bar in your applications is included in estools.h, demotool.c and demotool.rc. The overview is given below in this file.

Installation.

If you read this message, I assume you already uncompressed these files and I do not need to explain how to use pkunzip.exe. To use the DEMOTOOL.EXE just run it from MSWin 3.x. Make sure that estools.dll is in the same directory as demotool.exe or in your windows\system directory. You also should not rename the DLL (at least before you know how it works). Although I have not used any 3.1 specific APIs in the DLL, I tested it with 3.1 only and I do not know if it works with 3.0. I think it should. It should work on Chicago as well (as a 16 bit app.).

Overview.

Actually there is not much to overview.
The library exports 14 functions (2.01 version), some as Pascal and case insensitive, some as CDECL and case sensitive.

ToolBar functions:

CreateToolBar	@1
DeleteToolBar	@2
ESToolBarVers	@3
GetButtonNumber	@4
CreateToolBarIndirect	@5
LoadToolBar	@6
FreeToolBar	@7
InsertButton	@14

ToolBar messages

TBN_CHANGED

TBM_SETBTNSTATE
TBM_GETBTNSTATE
TBM_SETBTNSTYLE
TBM_GETBTNSTYLE
TBM_SETTBSTYLE
TBM_GETTBSTYLE

ToolBar Resource Script

StatusBar functions:

CreateStatusBar	@8
PostText	@9
PrintText	@10
PostTextRes	@11
PrintTextRes	@12
DoneStatus	@13

The program was written with Borland® C++ 3.1 & 4.02 and includes original project file. To compile with BC++ just load the project, make sure that the directories are right for your configuration and run. I do not use MS stuff (VC++), it must be compatible but I did not try it. People told me that it works OK.

If I am encouraged to write a next versions of the DLL I would like to include a way to remove buttons at run time, perhaps give a possibility to assign a custom rectangle for each button and make it possible to group a few buttons together (you can do this now by simply including a few TBars into a dialog box). I am also thinking about adding fly-by help to the Toolbar. Right now I am making a tree listbox control, just like the one Program Manager in MSW 3.1 uses except that the user will be able to add own bitmaps for items. Let me know if you have any suggestions.

Tool Bar.

The detailed description of these functions is given in estools.h. Tool Bar can be created in a number of styles, with caption or without, with border or without, it can be of **WS_CHILD** or **WS_POPUP** styles. It can be vertical, horizontal or square or whatever programmer wants. All styles (except child/popup) can be changed at run time. Buttons can be added at run time (just like menu items). Demotool.exe demonstrates all of this. Buttons may have 3 styles - (a) standard windows-like graphic button, (b) button which does not pop up until another button in the same TB is pressed (auto 2 state style) and (c) 2 state style when the button remains in the pressed state until it receives a message to pop up (2 state style). In addition buttons might be initially pressed, disabled (shaded) or enabled. Each button in the moment when it changes state sends a notification message to the parent. You can change all these button styles freely at run time.

Tool Bar Styles

TBS_BORDER
TBS_CHILD
TBS_FIXED
TBS_MOVABLE
TBS_NOBORDER
TBS_POPUP
TBS_VISIBLE

Tool Bar Button Styles

TBB_2STATE
TBB_AUTO2STATE
TBB_DISABLED
TBB_PRESSED
TBB_STANDARD

Resource Script

```
#include "estools.h"

/* Bitmap placed on top of TB buttons */
BTNS BITMAP "btns.bmp"

/* Bitmaps for buttons added at run time */
BMP0 BITMAP "bmp0.bmp"
BMP2_5 BITMAP "bmp2_5.bmp"
BMP6 BITMAP "bmp6.bmp"

/*ToolBar window caption */
STRINGTABLE
BEGIN
        NAME1, "ES ToolBar"
END

/* Script for the Tool Bar itself */
TOOLBAR RCDATA
BEGIN
    TB_RESOURCE_VERSION, /* Resource version, old resources will not work !!! */
    NAME1,                          /* ToolBar window caption */
    TBS_CHILD | TBS_MOVABLE | TBS_BORDER   |   TBS_VISIBLE,
                                    /* ToolBar style */
    BTNS,                           /* Bitmap with button faces */
    24, 24,                         /* Size of the individual face (width, height) */
    40, 32,                         /* button size - width, height */
    3,                              /* ToolBar border size -- ignored if no
                                        TBS_BORDER style specified*/
    5,                              /* Number of buttons in the horizontal line */
    5,                              /* Total number of controls in the ToolBar*/
    0, 0, ID_CMD1, TBB_DISABLED,
                                    /* controls (buttons) in the form:
                                            x-offset of the button's face in BTNS bitmap, y-offset,
                                            Button ID (wParam in WM_COMMAND
                                            message),
                                            Button style.
                                    */
    0, 24, ID_CMD2, TBB_STANDARD,
    0, 48, ID_CMD3, TBB_AUTO2STATE | TBB_PRESSED,
    0, 72, ID_CMD4,   TBB_STANDARD,
    0, 96, ID_CMD5, TBB_2STATE,
    0                                               /* not required, but recommended for future
                                        compatibility */
END
```

See also:

TBRESOURCEHEADER
TBCONTROLSTRUCT

Status Bar.

Exports @8-@13 are related to Status Bar. Please pay attention that exports 9-12 are CDECL functions. To link properly with the DLL you need to use import library estools.lib (use the one included or make your own). Everything about this part of the library is very much straight forward. _Post/_PrintText functions can digest anything what **wsprintf( )** can. The only difference between these two functions is that if you call _Print... it does not return until the text was actually printed on the SB window. _Post simply posts a message with the text. There might be a delay between the call to this function and actual output. To destroy the Status Bar window you can use a usual call to **DestroyWindow**, SB does not allocate memory or anything like that. You can also use any APIs or messages to manipulate it. It is a plain normal **WS_CHILD** window. **DoneStatus** draws a bar indicator with optional percent of the completed task. Check out demotool.exe.

About Author.

The author is Eugene L. Sokolov, third year grad.student at
the Department of Chemistry, SUNY at Stony Brook, NY.

My address is:
Eugene Sokolov,
Dept. of Chemistry,
SUNY at Stony Brook,
Stony Brook, NY 11794-3400
USA.
day time phone (516)632-7892,
Internet esokolov@sbchm1.chem.sunysb.edu

Comments and suggestions are welcome.

Help Contents

To display a list of topics by category, click any of the contents entries below. To display an alphabetical list of topics, choose the Index button.

C Elements

Other

Help file built: 12/04/94

About Autodoc

The sources for this Help file were generated by Autodoc, the source code documentation tool that generates Print or Help files from tagged comments in C, C++, Assembly, and Basic source files.

Autodoc is located on \\PALE\PUBLIC\AUTODOC. For information, contact Eric Artzt (erica@microsoft.com).

Functions

Messages

SBS_NONUMBERS
SBS_NUMBERS
TBB_2STATE
TBB_AUTO2STATE
TBB_DISABLED
TBB_PRESSED
TBB_STANDARD
TBM_GETBTNSTATE
TBM_GETBTNSTYLE
TBM_GETTBSTYLE
TBM_SETBTNSTATE
TBM_SETBTNSTYLE
TBM_SETTBSTYLE
TBN_CHANGED
TBS_BORDER
TBS_CHILD
TBS_FIXED
TBS_MOVABLE
TBS_NOBORDER
TBS_POPUP
TBS_VISIBLE

Structures and Enums

TBCONTROLSTRUCT
TBRESOURCEHEADER

Overviews

Modules

ERROR.C
ESTOOLS.C

Module ERROR.C

Description
Module containing Status Bar functionality

Module ESTOOLS.C

Description
Module containing the Tool Bar functionality.

CreateStatusBar

HWND WINAPI CreateStatusBar(DWORD *dwStyle***, int** *x***, int** *y***, int** *nWidth***, int** *nHeight***, HWND** *hwndParent***, HINSTANCE** *hInst***)**

Creates a StatusBar window.

Defined in: ESTOOLS.H

Return Value

Window handle of the StatusBar on success, NULL otherwise.

Parameters

dwStyle
Window style, directly passed to **CreateWindow** function;

x
Horizontal position of the left upper corner of the window;

y
Vertical position of the left upper corner of the window;

nWidth
Window width;

nHeight
Window height;

hwndParent
Parent window of the StatusBar;

hInst
Instance handle (must be an application instance, NOT library);

CreateToolBar

HWND CALLBACK CreateToolBar(HINSTANCE *hInst*, **LPCSTR** *lpszTemplate*, **HWND** *hwndParent*, **int** *x0*, **int** *y0*)

Creates a tool bar control from resource.

Defined in: ESTOOLS.H

Return Value

On sucsess returns a window handle of the ToolBar control, NULL otherwise.

Parameters

hInst

Instance handle (must be an instance of application, NOT library);

lpszTemplate

Points to a null-terminated string which contains the name of the ToolBar template.

hwndParent

Parent window of the ToolBar control.

x0

Specifies the initial x-position of the window. It is the x-coordinate of the upper-left corner of the window in the client area of its parent window.

y0

Specifies the initial y-position of the window. It is the y-coordinate of the upper-left corner of the window in the client area of its parent window.

CreateToolBarIndirect

HWND CALLBACK CreateToolBarIndirect(HINSTANCE *hInst*, **HTOOLBAR** *htbTemplate*, **HWND** *hwndParent*, **int** *x0*, **int** *y0*)

Creates a ToolBar control from a loaded resource.

Defined in: ESTOOLS.H

Return Value
On sucsess returns a window handle of the ToolBar control, NULL otherwise.

Parameters
hInst
Instance handle (must be an instance of application, NOT library);

htbTemplate
Handle returned by LoadToolBar function.

hwndParent
Parent window of the ToolBar control.

x0
Specifies the initial x-position of the window. It is the x-coordinate of the upper-left corner of the window in the client area of its parent window.

y0
Specifies the initial y-position of the window. It is the y-coordinate of the upper-left corner of the window in the client area of its parent window.

DeleteToolBar

BOOL CALLBACK DeleteToolBar(HWND *hwnd***)**

Destroys the tool bar. This function is called when the ToolBar window receives WM_DESTROY message.

Defined in: ESTOOLS.H

Return Value

On sucsess returns TRUE, FALSE otherwise.

Parameters

hwnd

Window handle of the TB control to be destroyed

DoneStatus

UINT WINAPI DoneStatus(HWND *hwndStatus*, **UINT** *nFlags*, **DWORD** *nDone*, **DWORD** *nTotal*)

Displays an indicator of action completion. It fills the client area of the SB with a COLOR_ACTIVECAPTION brush to the extend of nDone/nTotal and optionally displays the 100*nDone/nTotal %.

Defined in: ESTOOLS.H

Return Value
On success 100*nDone/nTotal. NULL otherwise.

Parameters
hwndStatus
Handle of the SB window;

nFlags
Flags indicating if the percent numbers have to be drawn.

SBS_NUMBERS
Draw the numbers;

SBS_NONUMBERS
Do not draw the numbers (just a bar).

nDone
How much of the action is complete;

nTotal
Total amount of work in the action;

Comments
If nDone>nTotal or nTotal==0, no action is taken.

ESToolBarVers

UINT CALLBACK ESToolBarVers(void)

DLL version. Retrieves a version number of the library.

Defined in: ESTOOLS.H

Return Value

DLL version number, in hexadecimal. For example for the current version (2.01), this function returns 0x201.

FreeToolBar

VOID CALLBACK FreeToolBar(HTOOLBAR *htbTemplate***)**

Frees loaded ToolBar resource.

Defined in: ESTOOLS.H

Return Value
None

Parameters
htbTemplate
Handle of the ToolBar resource to be freed (loaded by **LoadToolBar( )**);

See Also
LoadToolBar

GetButtonNumber

int CALLBACK GetButtonNumber(HWND *hwnd*, **UINT** *nId*)

Retrieves the button number from it's ID number, analogous to **GetDlgItem** except that the TB buttons are not individual windows and consequently do not have handles.

Defined in: ESTOOLS.H

Return Value

Button number, which can be used to change button's state or style. If no button with such ID exists function returns -1. If several buttons share the same ID it returns first it encounters.

Parameters

hwnd

ToolBar window handle;

nId

Button ID (equal to the wParam of WM_COMMAND from the corresponding button, or second parameter (tbsMsg) in the ToolBar resource).

See Also

TBCONTROLSTRUCT

TBRESOURCEHEADER

InsertButton

UINT CALLBACK InsertButton(HWND *hwnd*, **UINT** *id*, **UINT** *nFlags*, **UINT** *idNewId*, **HBITMAP** *hbmpNewFace*)

Inserts a new button into the ToolBar moving other buttons down the bar.

Defined in: ESTOOLS.H

Return Value

Button number of newly added button, which can be used to change button's state or style. If failed returns (UINT)-1.

Parameters

hwnd

TB window handle;

id

Specifies the TB item before which the new menu item is to be inserted, as determined by the nFlags parameter.

nFlags

Specifies how the id parameter is interpreted and information about the state of the new ToolBar item when it is added to the ToolBar. This parameter consists of one of the following values.

MF_BYCOMMAND

The id parameter specifies the TB-item identifier.

MF_BYPOSITION

The id parameter specifies the zero-based position of the TB item (button number). If id is (UINT)-1, the new TB item is appended to the end of the TB.

idNewId

Specifies either the identifier of the new TB item.

hbmpNewFace

Specifies the bitmap handle of the new TB item.

LoadToolBar

HTOOLBAR CALLBACK LoadToolBar(HINSTANCE *hInstance*, **LPCSTR** *lpszTemplate*)

Loads a ToolBar resource.

Defined in: ESTOOLS.H

Return Value
Handle of the loaded ToolBar resource on success, NULL otherwise.

Parameters
hInstance
Identifies an instance of the module which executable file contains the ToolBar resource to be loaded.

lpszTemplate
Points to a null-terminated string which contains the name of the ToolBar template.

Comments
If TB resource was loaded but was not used to create a window, the handle must be freed by calling **FreeTolBar** function. If ToolBar window was successfully created using this handle the handle was automatically deleted.

PostText

UINT FAR CDECL PostText(HWND *hwndStatus*, **LPSTR** *lpszFormat*, *...*)

Formats and prints series of characters and values on the StatusBar by sending WM_SETTEXT message to the StatusBar window. Each argument (if any) is converted according to the corresponding format specified in the format string (through **wsprintf()** function). This function is the best way to output with StatusBar.

Defined in: ESTOOLS.H

Return Value
On success the number of bytes printed. NULL otherwise.

Parameters
hwndStatus
Handle of the SB window;

lpszFormat
Address of format-control string;

...
Specifies zero or more optional arguments;

Comments
see details on **wsprintf()**;

See Also
PrintText

PostTextRes

PostTextRes

UINT FAR CDECL PostTextRes(HWND *hwndStatus*, **HINST** *hInst*, **int** *nStr*)

Same as PostText, except it loads a format string nStr from hInst module;

Defined in: ESTOOLS.H

Return Value
On success the number of bytes printed. NULL otherwise.

Parameters
hwndStatus
Handle of the SB window;

hInst
Instance handle where the format string resource is located;

nStr
Integer identifier of the string to be loaded;

Comments
String has to be shorter than 128 bytes;

See Also
PrintTextRes

PrintText

UINT FAR CDECL PrintText(HWND *hwndStatus*, **LPSTR** *lpszFormat*, *...*)

Formats and prints series of characters and values on the SB by painting them directly onto the device context. Each argument (if any) is converted according to the corresponding format specified in the format string (through **wsprintf** function).

Defined in: ESTOOLS.H

Return Value
On success the number of bytes printed. NULL otherwise.

Parameters
hwndStatus
Handle of the SB window;

lpszFormat
Address of format-control string;

...
Specifies zero or more optional arguments;

Comments
see details on **wsprintf()**; Use this function only if PostText does not produce a desired result. (like in the case of real-time mouse tracking).

See Also
PrintTextRes

PrintTextRes

UINT FAR CDECL PrintTextRes(HWND *hwndStatus*, **HINSTANCE** *hInst*, **int** *nStr*)

Same as PrintText, except it loads a format string nStr from hInst module;

Defined in: ESTOOLS.H

Return Value
On success the number of bytes printed. NULL otherwise.

Parameters
hwndStatus
Handle of the SB window;

hInst
Instance handle where the format string resource is located;

nStr
Integer identifier of the string to be loaded;

Comments
String has to be shorter than 128 bytes;

See Also
PostTextRes

SBS_NONUMBERS

Do not draw the numbers

Defined in: ESTOOLS.H

SBS_NUMBERS

Draw the numbers

Defined in: ESTOOLS.H

TBB_2STATE

Button remains depressed until it state is changed by sending it a message TBM_CHANGEBTNSTATE

Defined in: ESTOOLS.H

TBB_AUTO2STATE

Button remains depressed untill another button in the same tool bar is pressed;

Defined in: ESTOOLS.H

TBB_DISABLED

Button disabled (shadowed)

Defined in: ESTOOLS.H

TBB_PRESSED

Button is initially pressed

Defined in: ESTOOLS.H

TBB_STANDARD

Standard Windows-like button -- default;

Defined in: ESTOOLS.H

TBM_GETBTNSTATE

Get state of the button;

Defined in: ESTOOLS.H

Return Value
Button state (TRUE == the button is pressed).

Parameters
wParam
Button number

lParam
Unused

TBM_GETBTNSTYLE

Get button style.

Defined in: ESTOOLS.H

Return Value
WORD Button style;

Parameters
wParam
Button number

lParam
Unused

TBM_GETTBSTYLE

Get ToolBar style and a number of buttons per row.

Defined in: ESTOOLS.H

Return Value
ToolBar style in LOWORD, Number of buttons per row in HIWORD

Parameters
wParam
Unused

lParam
Unused

TBM_SETBTNSTATE

Sets the state of the button

Defined in: ESTOOLS.H

Return Value
(BOOL)Button state (TRUE == Press the button).

Parameters
wParam
Button number

lParam
LOWORD(lParam)==state

TBM_SETBTNSTYLE

Sets button stule

Defined in: ESTOOLS.H

Return Value
Old button style;

Parameters
wParam
Button number
LOWORD(lParam)
New button style

TBM_SETTBSTYLE

Sets tool bar style

Defined in: ESTOOLS.H

Return Value
Old ToolBar style in LOWORD, Number of buttons per row in HIWORD

Parameters
wParam
contains the same values as a wStyle field in TBRESOURCEHEADER,

LOWORD(lParam)
shold contain the new number of buttons per row, if it is 0, then it is ignored. Unlike the resource header where 0 in this field means maximum possible number of controls in a row.

TBN_CHANGED

Notifies that the button has changed its state.

Defined in: ESTOOLS.H

Parameters

wParam

button number,

lParam

LOWORD(lParam) = 1 if the button was pressed, 0 otherwise (the state was changed through

See Also

TBM_SETBTNSTATE message.

HIWORD(lParam) = TRUE if changed by mouse click (not by moving the cursor into or out of button area)

TBS_BORDER

ToolBar has a border around controls. If this bit is clear, than value nBorder in TBRESOURCEHEADER is ignored;

Defined in: ESTOOLS.H

TBS_CHILD

Translated into WS_CHILD -- default;

Defined in: ESTOOLS.H

TBS_FIXED

No caption -- ignored if TBS_POPUP set;

Defined in: ESTOOLS.H

TBS_MOVABLE

Has caption -- default;

Defined in: ESTOOLS.H

TBS_NOBORDER

No border around controls -- default;

Defined in: ESTOOLS.H

TBS_POPUP

Translated into WS_POPUP;

Defined in: ESTOOLS.H

TBS_VISIBLE

Translated into WS_VISIBLE;

Defined in: ESTOOLS.H

TBCONTROLSTRUCT

```
typedef struct {
    int tbcCX;
    int tbcCY;
    int tbcMsg;
    int tbcStl;
} TBCONTROLSTRUCT;
```

Resource for each individual button

Defined in: ESTOOLS.H

Members

tbcCX
X-offset in bitmap;

tbcCY
Y-offset in bitmap;

tbcMsg
Button ID

tbcStl
Button style

TBRESOURCEHEADER

```
typedef struct {
    WORD nVersion;
    WORD nWndName;
    WORD wStyle;
    WORD nBitmap;
    int nXBmp;
    int nYBmp;
    int nXSize;
    int nYSize;
    int nBorder;
    int nRowLen;
    int nCtrl;
    TBCONTROLSTRUCT tbcsCtrl[1];  //  MQN;
} TBRESOURCEHEADER;
```

Structure containing Tool Bar resource

Defined in: ESTOOLS.H

Members

nVersion
Resource version number. Added for future compatibility. Has to be set to 0xE100 (TB_RESOURCE_VERSION)

nWndName
String ID -- identifies TB's name

wStyle
TooBar style

nBitmap
Bitmap ID (with button faces);

nXBmp
Width of individual bitmap on each button

nYBmp
Height ...

nXSize
Button width

nYSize
Button height

nBorder
Border size

nRowLen
Number of buttons per horozontal row.

nCtrl
Total number of buttons.

MQN
Individual buttons.

License.

This is a demo version of Tool Bar and Status Bar library (the Software). You can freely use Software for demonstration purposes. Software can be redistributed as long as all files listed above are included in the distribution package in the original form. No fee can be taken for distribution of the Software except media and transmission costs.You can freely use or edit all source code (demotool.c, esdefs.h, estools.h, estools.def, demotool.rc) included with Software as long as the original copyright notice inside these files remains unaltered. No permission granted to change or reverse engineer by any means the **estools.dll** module.

Although all considerable effort was spent to make the Software effective and bug free, no warranties are given. In no event Author (whose name and address are given below) shall be liable for any direct or indirect damage arising from use or inability to use the Software.

Registration.

Now, the most interesting part. I would gratefully appreciate if you register. This DLL is not of much use by itself, it is good as a part of some software package. If you register you
(a) get a **clean estools.dll** (no message boxes in the beginning), permission to redistribute the DLL (non-commercial, if you want to redistribute them commercially, write me);
(b) it costs US$25 only (students $15).

To register print out the order form from order.wri (you may print on both sides, first page is the order form, second is my address, than fold the paper and you do not need an envelope) include payment, seal and mail. Or do it any way you want, just make sure your name and address are clearly readable.