

Тема 8. Проектирование интерфейса окна

© 2011-2012

Строка состояния

Строка состояния (status bar) — это окно, обычно располагающееся в нижней части главного окна приложения, которое предназначено для информирования пользователя о текущем состоянии программы, о выполняемых операциях и режимах.

Для использования в программе строки состояния нужно выполнить дополнительные подключения

```
#pragma comment(lib, "comctl32.lib")  
#include <commctrl.h>  
  
#include <commdlg.h>
```

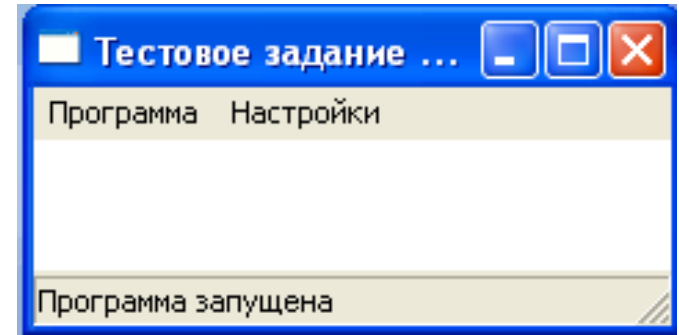
Создание строки состояния

Для создания строки состояния следует использовать функцию

```
HWND WINAPI CreateStatusWindow(  
    LONG style,  
    //стиль: обязательно указываем WS_CHILD | WS_VISIBLE  
    LPCTSTR lpszText, // текст в строке состояния  
    HWND hwndParent,  // родительское окно  
    UINT wID           // числовой id строки состояния  
);
```

Создание строки состояния

Пример



```
HWND hStatus; // глобальная переменная
```

```
// задаем в WinMain
```


```
hWnd=CreateWindow(szClassName,szWindowTitle,  
    WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,  
    CW_USEDEFAULT, CW_USEDEFAULT,0,0,0,hInst,0);
```

```
. . .
```

```
hStatus = CreateStatusWindow(WS_CHILD | WS_VISIBLE,  
    TEXT("Программа запущена"), hWnd, 10000);
```

```
. . .
```

```
ShowWindow(hWnd,nCmdShow);  
UpdateWindow(hWnd);
```



Размеры и позиция строки состояния

Оконная процедура дочернего окна строки состояния автоматически устанавливает начальную позицию и размеры этого элемента управления.

Ширина строки состояния равна ширине клиентской области родительского окна.

Высота строки состояния устанавливается на основе метрик шрифта, выбранного по умолчанию в контекст устройства элемента управления.

Размеры и позиция строки состояния

При каждом изменении размеров родительского окна, то есть при получении сообщения `WM_SIZE`, оконная процедура `WndProc` должна отправить строке состояния такое же сообщение, передав текущие значения параметров `wParam` и `lParam`:

```
case WM_SIZE:  
    SendMessage (hStatus, WM_SIZE, wParam, lParam);  
    break;
```

Изменение высоты строки:

```
SendMessage (hwndStatusBar, SB_SETMINHEIGHT,  
             minHeight, 0);
```

где `minHeight` — минимальная высота окна строки состояния в пикселах.

Режимы строки состояния

- *Стандартный, или многочастный, режим (Multiple-Part Status Bars)*, в котором строка состояния разбивается на несколько частей (полей). В каждом поле выводится отдельная строка текста.
- *Простой режим (Simple Mode Status Bars)*, в котором строка состояния реализована как единый элемент и отображает только одну строку.

Переключение между режимами осуществляется посылкой сообщения `SB_SIMPLE`:

```
SendMessage (hwndStatusBar, SB_SIMPLE, fMode, 0);
```

где `fMode = TRUE`, - простой режим,

`FALSE` - стандартный (многочастный) режим.

Вывод текстового сообщения

1 способ

С помощью функции изменения текста указанного окна:

```
BOOL SetWindowText(  
    HWND hWnd,           // дескриптор окна  
    LPCTSTR lpString     // указатель на новую строку  
);
```

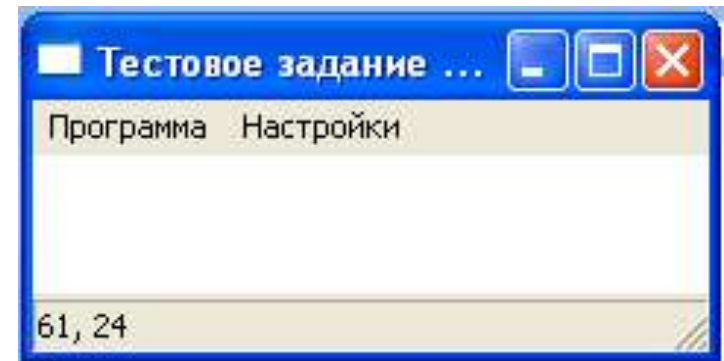
Пример:

отображение позиции мыши

```
int MouseX, MouseY;  
static TCHAR stext[20];  
...
```

```
case WM_LBUTTONDOWN:
```

```
    MouseX = GET_X_LPARAM(lParam);  
    MouseY = GET_Y_LPARAM(lParam);  
    swprintf(stext, 20, L"%d, %d", MouseX, MouseY);  
    SetWindowText(hStatus, stext);  
    break;
```



Вывод текстового сообщения

2 способ

С помощью отсылки сообщения `SB_SETTEXT` с помощью функции:

```
SendMessage(hStatus, SB_SETTEXT,  
            wParam, (LPARAM) szText);
```

где `szText` — указатель на C-строку,
`wParam` — задает номер поля в строке состояния и графический стиль для этого поля; для простого режима этот номер = 0.

Разделение строки состояния на поля

1 шаг: С помощью отсылки сообщения SB_SETPARTS :

```
SendMessage(hStatus, SB_SETPARTS,  
            nParts,           // количество частей  
            (LPARAM) aWidths); // указатель на массив  
                               //размеров частей
```

2 шаг: С помощью отсылки сообщения SB_SETTEXT:

```
SendMessage(hStatus, SB_SETTEXT,  
            iPart,           // номер части  
            (LPARAM) szText); // текст
```

Разделение строки состояния на поля (пример)

```
int widths[5]={100,150,200,-1};
```



WinMain

```
hStatus = CreateStatusWindow(WS_CHILD | WS_VISIBLE |  
    SBARS_SIZEGRIP, "text", hwnd, 100);
```

WndProc

```
case WM_CREATE:
```

```
    SendMessage(hw, 100, SB_SETPARTS, 4, (LPARAM)widths);  
    SendMessage(hw, 100, SB_SETTEXT, 2, (LPARAM) "part 2");  
    SendMessage(hw, 100, SB_SETTEXT, 3, (LPARAM) "last part");  
    ...
```

В диалоговом окне используется функция `SendDlgItemMessage`.

Элемент управления “BUTTON”

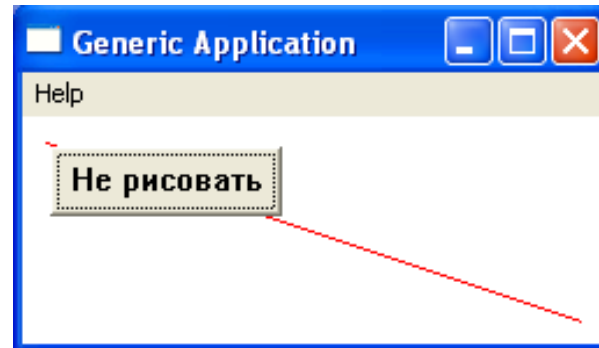
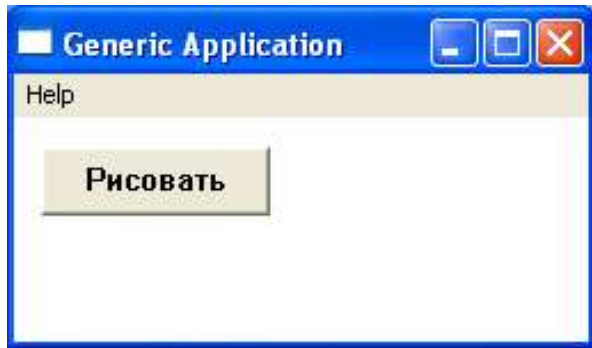
Кнопка - это небольшое прямоугольное дочернее окно, обычно имеющее два состояния: нажато/отпущено или включено/выключено.

Пользователь меняет состояние этого элемента щелчком мыши.

К этому классу относятся:

- кнопки-"давилки" (*push buttons*),
- кнопки-"галочки" (*check boxes*),
- "радио"-кнопки (*radio buttons*)
- специальный тип групповых рамок (*group boxes*).

Пример работы с кнопками



```
// создание элементов управления
#define ID_MYBUTTON 3001
bool draw_flag = false; // рисуем или нет
HWND hMyButton;

void UpdateMyButton() {
    if (draw_flag)
        SetWindowText(hMyButton, "Не рисовать");
    else
        SetWindowText(hMyButton, "Рисовать");
}
```

Пример работы с кнопками

```
switch( msg ) {
case WM_CREATE:
    hMyButton = CreateWindow("Button", "AAA", WS_CHILD |
        WS_VISIBLE | BS_PUSHBUTTON,
        12, 12, 100, 30, hWnd,
        HMENU) ID_MYBUTTON, hInstance, NULL);

    ...
    return 0 ;
case WM_COMMAND:
    switch( wParam ) {
    case ID_MYBUTTON:
        draw_flag = !draw_flag;
        UpdateMyButton();
        RedrawWindow(hWnd, NULL, NULL,
            RDW_INVALIDATE | RDW_ERASE);

        break;
    }
    break;
```



Спасибо за внимание!