Index

Ordering Storc

Thank you for trying out this demo!

We hope it gives you a good idea of what STORC GOLD is capable of doing for you.

The program you are looking at has all the features of STORC GOLD as of June 30, 1993. We constantly improve the product and add useful features. Please consider ordering the full version of STORC GOLD (See Order Form)

Here are some typical uses for Storc (check all that applies):

- Convert your VB form into SDK resource script
- Convert your VB form into Visual C++ compatible resource script with VBX controls
- Convert a form created by using another prototyping tool into SDK
- Import a form from your competitor's product. See what kind of custom controls they used
- Capture the Main Menu and Sysmenu of any running application. Some applications modify the menus -- or even create them at runtime. With Storc, you can capture current menu state
- Capture background brush and color information for every control
- Capture the font information from any window

We wish you success in your work!

Basics
About this demo
About PractiSys
Product Support
Order Form

Product Support

We will support this product through CompuServe, by Fax, or by Mail for 3 full months from the date of purchase.

About PractiSys

PractiSys specializes in practical tools for Microsoft Windows environment.

Our address is :

PractiSys

4767 Via Bensa
Agoura, CA 91301
U.S.A.

Fax(orders/support): (818) 706-8877
CompuServe: 72010,567
E-MAIL: 72010.567@COMPUSERVE.COM

PractiSys ◆ STORC GOLD Order Form ◆ 818-706-8877

..... Please, select 'File / Print Topic' to print out this Form .....

Name [ ] Company [ ]
Street Address [ ]
City [ ] State [ ] Zip[ ]
CompuServe ID [ ] Disk Size: [ ]5.25" [ ]3.5"
Payment Method:
[ ] Check
[ ] C.O.D. (no extra charge).
How did you hear about your products? [ ]

Product	Price	Number of Copies	Subtotal
Storc Gold 1.0	39.93		
		CA State Tax	
		Shipping/Handling[1]	
		Total for this order	
[1] U.S. and Canada: $6, International: $15 (per copy)			

We offer 30 day money-back guarantee, so if you decide you cannot use STORC for any reason, your money will be cheerfully refunded.

PAYMENT OPTIONS

- Check
- C.O.D.
- CompuServe
- Visa/Mastercard
- International Orders

All orders ship 1st class mail.

A check or money order in U.S. funds drawn on a U.S. bank.
We typically DO NOT wait for checks to clear.
(Non-U.S. customers: please see International Orders for details)

Fill out this form and mail with payment to:

PractiSys
4767 Via Bensa
Agoura, CA 91301

C.O.D. (Continental U.S. and Canada only)
NO SURCHARGE!

Fill out the order form. Make sure the address is street address (no POB)
Do one of the following:

Mail to:

PractiSys
4767 Via Bensa
Agoura, CA 91301

or fax to: PractiSys (818) 706-8877 (9-5 PST)

You can also send a message with your street address (no POB) to us at

CompuServe: 72010,567 or
Internet: 72010.567@compuserve.com

GO SWREG
Register STORC1.EXE (registration ID is 664). CompuServe will bill you the usual way. We will receive a notification from CIS as soon as you register, and promptly ship the software.

Sorry, we do not accept credit cards. Ordering through CompuServe offers similar protection and fast turnaround. You can also order C.O.D. at no extra charge (please see Ordering through CompuServe and Ordering C.O.D.).

Shipping/handling charge is $15.
Ordering through CompuServe(SWREG) is best. Please e-mail to 72010,567 for instructions on how to pay additional shipping. You can also pay by check, but it has to be in U.S. funds drawn on a U.S. bank. We will discuss any other payment option you consider convenient.

STORC Basics

STORC is a universal form conversion tool!

Take a VB form and convert it to a DIALOG script. Port your whole VB prototype to SDK! Reuse your VBX controls under Visual C++! Capture Menus. Reverse-engineer forms. Figure out background brush used to paint a control -- or which font is used in this button...

STORC! You are running a fully-functional demo...

More About Storc

Any window on screen, regardless of the way it was created can be represented in Windows resource format. If you have a project with forms created in a nonstandard way, STORC is a natural choice to 'extract' its form design and reuse the forms in another project.

With the diversity of form design packages (all with different form representation) on the market these days, STORC provides you an easy conversion to tried and true Windows DIALOG script.

There are special features aiding in conversion of Visual Basic forms. If you are going to use Microsoft Visual C++, all VBX controls can be reused or mapped into other VBX (or non-VBX) controls.

Storc is based on a standard text editor, so you will find most of the familiar menu choices.

There is an additional menu choice (Form/Find) that allows you to select your target (form) window. As soon as this is done, STORC reads the window and creates a script that is displayed in the editor.

Please check out how Storc imports menu, color and font information!

About this Demo

This demo shows actual STORC interface and output, allowing you to make the buying decision.

However, it does not allow saving the resulting script.

If you want to make use of all the features, need better help and user manual, please purchase the actual product.

Credits

DIALOG Statement

Windows Resource Compiler 'DIALOG' Statement

The 'DIALOG' statement is used in Windows resource (*.rc) files to describe a dialog box. The statement defines the position and dimensions of the dialog box on the screen as well as the dialog box style.

Here is general format of the DIALOG statement:

nameID DIALOG [load-option] [mem-option] x , y , width , height
STYLE Style
CAPTION Caption
MENU Menu
CLASS Class
FONT FontSpec
BEGIN
    control-statement s
    .
    .
    .
END

nameID

Identifies the dialog box. This is either a unique name or a unique integer value in the range 1 to 65,535.

LoadOption

Specifies when the resource is to be loaded. This parameter is optional. If it is specified, it must be one of the following:

PRELOAD
Resource is loaded immediately.

LOADONCALL
Resource is loaded when called. This is the default option.

MemOption

Specifies whether the resource is fixed or movable and whether it is discardable. This parameter is optional. If it is specified, it must be either FIXED or MOVEABLE. An additional value, DISCARDABLE may also be specified.

FIXED
Resource remains at a fixed memory location.

MOVEABLE
Resource can be moved if necessary in order to compact memory. This is the default option.

DISCARDABLE
Resource can be discarded if no longer needed.

x

Specifies the x-coordinate of the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The coordinate is assumed to be in dialog units. Can either be relative to the window's parent or owner window, or relative to the origin of the screen. This is determined by the window's style setting. Dialog windows are positioned relative to their parent or owner window unless the dialog window style constant DS_ABSALIGN is used.
Dialog Units

y

Specifies the y-coordinate of the top side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The coordinate is assumed to be in dialog units. Can either be relative to the window's parent or owner window, or relative to the origin of the screen. This is determined by the window's style setting. Dialog windows are positioned relative to their parent or owner window unless the dialog window style constant DS_ABSALIGN is used.

Dialog Units

width

Specifies the width of the dialog box. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The width is in 1/4-character units.

Dialog Units

height

Specifies the height of the dialog box. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The height is in 1/8-character units.

Dialog Units

style

Specifies the dialog box styles. This parameter can be one of the following values:

DS_LOCALEDIT
DS_MODALFRAME
DS_SYSMODAL

Also any window styles (WS_) can be used.

If you do not specify the style, the default is WS_POPUP | WS_BORDER | WS_SYSMENU

WS_BORDER

Creates a window that has a border

WS_CAPTION

Creates a window that has a title bar (implies the WS_BORDER style)

WS_CHILD

Creates a child window. Incompatible with the WS_POPUP style

WS_GROUP

For controls only: specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls defined without the WS_GROUP style after the first control belong to the same group. The next control with the WS_GROUP style starts the next group

WS_POPUP

Creates a pop-up window. Incompatible with the WS_CHILD style

WS_SYSMENU

Creates a window that has a System-menu box in its title bar. Used only for windows with title bars. If used with a child window, this style creates a Close box instead of a System-menu box.

DS_LOCALEDIT

Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature may be suppressed by adding the DS_LOCALEDIT flag to the Style command for the dialog box. If this flag is not used, EM_GETHANDLE and EM_SETHANDLE messages must not be used, because the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes.

DS_MODALFRAME

Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the WS_CAPTION and WS_SYSMENU styles.

DS_NOIDLEMSG

Suppresses WM_ENTERIDLE messages that Windows would otherwise send to the owner of the dialog box while the dialog box is displayed.

DS_SYSMODAL

Creates a system-modal dialog box.

DS_ABSALIGN

If a dialog box template style sets DS_ABSALIGN flag, the dialog box coordinates (x, y) specified in the template are assumed to be relative to the top left corner of the screen. Otherwise(default) they are assumed to be relative to parent window of that dialog box.

Dialog Statement
Dialog Statement Usage Example
Dialog Units
Screen Related Coordinates Option

Caption

Specifies the dialog box caption - a text string in double quotes

Menu

The resource identifier or numeric ID of the associated menu. If the menu line is not present in the DIALOG definition, no menu is associated with the window.

Class

The class line in the DIALOG definition. It overrides the standard processing of a dialog window. The dialog window is assigned the specified class, rather than the standard dialog class. Class is an integer or text string that specifies the desired window class. Even though STORC can display the class field, we advise to never include it into your dialog templates, since it can be dangerous. By default, dialog windows are given a class that Windows implements. Using a custom dialog class provides additional control over the behavior of the dialog window, however this is hardly useful. In order to create a custom dialog class, you must set the cbWndExtra field of the WNDCLASS structure to at least as many bytes as used by DLGWINDOWEXTRA, the default dialog class.

FontSpec

The font specification. It consists of a point size (in pixels) followed by a font typeface string (for example, 14, "Helv"). Windows uses the bold weight for the font when this field is used. To use a lighter-weight attribute, use the WM_SETFONT message at runtime to set the font.

DialogUnits

The coordinates x, y, width, height should be given in DIALOG UNITS. The exact meaning of the coordinates depends on the style defined by the STYLE option statement and font for the dialog box. For child-style dialog boxes, the coordinates are relative to the origin of the parent window, unless the dialog box has the style DS_ABSALIGN; in that case, the coordinates are relative to the origin of the display screen. If a dialog box has the DS_ABSALIGN style, the dialog coordinates for its upper-left corner are relative to the screen origin instead of to the upper-left corner of the parent window. You would typically use this style when you wanted the dialog box to start in a specific part of the display no matter where the parent window may be on the screen.

Dialog Statement Usage Example

```
#include <windows.h>
MyDialog DIALOG   10, 10, 300, 200
STYLE WS_POPUP | WS_BORDER
CAPTION "How are you today?"
BEGIN
        CTEXT "Select One:", 1, 10, 10, 300, 20
        PUSHBUTTON "&Good!", 2, 150, 30, 60, 20
        PUSHBUTTON "&Ok!", 3, 150, 50, 60, 20
        PUSHBUTTON "&Fine!", 4, 150, 80, 60, 20
END
```

Dialog Script 'CONTROL' Statement

A Control-statement defines a control inside the dialog box. There are 2 ways of specifying controls: using the CONTROL keyword and using particular control type keyword.
You probably are used to the 2nd method (see the example above), but STORC uses the 1st one (the CONTROL statement), and for a good reason: this is the only statement with consistent parameters, for all the control-specific statements (like LTEXT, RTEXT, PUSHBUTTON, LISTBOX) parameters are type-dependent.

Control Statement Format

Control Statement

Here is the format of CONTROL statement:

CONTROL text , id , class , style , x , y , width , height The statement defines the position and dimensions of the control within the parent window as well as the control style.

VBX 'CONTROL' Statement

To support VBX controls, Microsoft has 'stretched' the CONTROL statement format. The old format now plays new tricks.
For a VBX control, the 'text' field contains information on VBX filename, control name, and control text. The 'class' field is fixed and must be 'VBControl'.
VBX Control Statement Format

VBX Control Statement

Here is the format of CONTROL statement for VC++ VBX controls:

CONTROL "vbxfile ;vbxcontrolname ;controltext ", id , "VBControl", style , x , y , width , height

The 'text' field contains information on VBX filename, control name and text itself.

VBX Control Text

This optional field contains window text for the VBX control.

VBX Control Statement Format

text

Specifies displayed text. Its position depends on the control class. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice. In the appropriate styles, an ampersand (&) character in the text indicates that the following character is used as a mnemonic character for the control. When the control is displayed, the ampersand is not shown, but the mnemonic character is underlined. The user can choose the control by pressing the key corresponding to the character.

id

Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.

Class

Specifies the control class. This value can be a pre-defined name, character string, or integer value that defines the class.

Style

Specifies the control style. The bitwise OR (|) operator is used to combine styles.

X

Specifies the x-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The coordinate is assumed to be in dialog units and is relative to the origin of the parent window.

y

Specifies the y-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The coordinate is assumed to be in dialog units and is relative to the origin of the parent window.

Width

Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The value is in 1/4-character units.

height

Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction () operator. The value is in 1/8-character units.

vbxfile

Specifies the filename containing the code/data that constitute a VBX control, e.g. 'THREED.VBX' The extension does not have to be .VBX, but it typically is.

vbxcontrolname

Specifies the name of a VBX control within a VBX file (a .VBX can contain several controls)
The prefix 'Thunder' is added to this name to make window class name for the control.
E.g. 'THREED.VBX' contains 'SSRibbon' control.

The specified control name MUST exist in the .VBX file.

Not to be confused with control text, which can be anything.

VBX Control Statement Format

Storc Capabilities

STORC creates a fully functional dialog template with CONTROL statements for every control in your form. It reads the dialog box and control text (if any), window class, style and font information, and interprets control styles for better readability. It allows you to tweak the way it treats control window classes. It has a little editor to help you modify what you want, too.
The output can be saved to a file or pasted into another editor, including a dialog editor like Borland Resource Workshop, and in most cases (with proper options selected) there will be nothing that you should do to make the resulting script run.

If you are using Microsoft Visual C++, you can reuse the VBX controls in a dialog box. With NativeVBX option enabled, STORC will create a dialog script in the special format required for VBX controls under VC++.

Note, that VC++ 1.0 only allows to reuse VBX controls compatible with Visual Basic 1.0. STORC allows you to remap one VBX control into another, or any control class into any other.

Working with STORC Application

Working with STORC Application

Suppose you have a Visual Basic prototype product with some forms.

Typically, you would run your demo and launch STORC.EXE.

To convert a form, you have to have it displayed on the screen, so STORC can read it.

When you see the form, activate STORC and select Form/Find from its main menu. The mouse cursor turns into a special arrow with words 'find window' inside and STORC minimizes itself so that you can see your form.

All you have to do now is click on the form window.

Note:
when you move the mouse above windows, window borders are highlighted to indicate which window will get selected if you click; we recommend clicking on Caption Bars for accuracy.

After you click, STORC restores itself with a resource script matching your form in its editor.

You can perform this operation for any number of forms; every time the resource script for the new form will be inserted into STORC editor at cursor, and you can cut/paste it to (or save it as) the RC file for your project

After you obtained the script, you can save it or cut and paste it into your resource editor.

For example, if you are working with Microsoft Visual C++:
--Run AppStudio and Workbench
--Create a new Dialog resource in AppStudio. Save resource script.
--Open your program's resource script as a text file in Workbench. There will be an empty dialog template
--Copy the whole dialog script from storc editor after the template. You've got 2 dialog templates now
--Change the dialog resource name on the template you pasted from Storc to the resource name AppStudio created for you (empty template).
--Delete the empty template.
--Open the resource file with AppStudio to see the resulting box.

Storc Output

The Storc Editor

The STORC application is built around an editor; its main window is actually an edit field, and it has 'file' menu, 'edit' menu and 'find' menu provide basic functionality of a 'normal' text editor.

Menus

Editor Menus

FileMenu
EditMenu
SearchMenu
OptionsMenu

File Menu

File/New Alt-F, N
Opens an 'untitled' text file

File/Open Alt-F, O
Displays a browse box and opens the (existing) file specified

File/Save Alt-F, S
Saves current buffer if you have already named the file; otherwise same as File/SaveAs

File/SaveAs Alt-F, A
Displays a browse box and saves editor buffer as a file with selected name

File/Exit Alt-F, X
exits STORC application

Edit Menu

Edit/Undo Alt-BkSp
undoes the last editing operation

Edit/Cut Shift-Del
copies selected text to clipboard and deletes selection

Edit/Copy Ctrl-Ins
copies selected text to clipboard, keeps selection

Edit/Paste Shift-Ins
pastes text from clipboard at cursor

Edit/Delete Del
deletes selected text

Edit/ClearAll Ctrl-Del
clears the editor buffer

Search Menu

Search/Find **Alt-S,F**
displays a 'search' box where you can specify a text string to search for. Search can be case-sensitive or case-insensitive. This command searches ONCE. To do another search of the same string from cursor, use Search/Next.

Search/Replace **Alt-S, R**
same as search/find, but allows you to replace the found string with specified text. Will search/replace once, unless 'All occurrences' is checked. If 'Prompt on replace' is checked, the editor will prompt you every time it finds a match. A 'Search/Replace: Replace this occurrence?' box will appear and found string will be highlighted. You can elect to replace the string found by selecting 'yes', search for next occurrence by pressing 'No' or stop search by pressing 'Cancel'.

Search/Next **Alt-S, N**
searches for the string specified by previous Search/Find or Search/Replace.

Options Menu

Translate Classes Option
Interpret Styles Option
Force Dialog Templates Option
Screen Related Coordinates Option
Native VBX Option

Translate Classes Option

You have to specify a CLASSNAME for every control in the DIALOG script. For example, you say
CTEXT "Text", -1, 7, 9, 75, 17, WS_CHILD | WS_GROUP
or (same thing):
CONTROL "Text", -1, "STATIC", WS_CHILD | WS_GROUP | SS_CENTER , 7, 9, 75, 17

Many of the tools define their own control styles; for example Visual Basic defines custom control classes like "ThunderLabel" or "ThunderListbox", etc. If you decide to convert a form with custom controls to resources and then run then from your program, you are likely to get into trouble. You either have to use the custom control DLLs you tool uses (and use them exactly as they were designed to be used) or provide a substitute for custom controls, mapping their classes into standard Windows control classes or your own classes. STORC 1.0 gives you both options, controlled by 'Options/TranslateClasses' menu item. When this menu item is checked, the control classes will be translated as defined in [class translation] section of STORC.INI file (described below). Otherwise you will see the original class names (which can be an educational experience).

control-statement
class

Interpret Styles Option

A window style is a LONG integer value associated with a window. It is logically divided into 2 words; high word contains WS-type styles shared by all window types. Low word is window-specific, and its meaning is dependent on window class.

If 'Options/InterpretStyles' is checked, STORC will make an attempt to interpret BOTH parts of window style; of course it only knows about styles of standard controls, so it will not interpret the lower word of the window style unless 'Options/TranslateClasses' is on and it could map the control class into one of the standard classes. It however would by all means attempt to interpret WS styles. Uncheck this option if you would like to see a long integer for style with no interpretation attempt.

Dialog Window Style
Control Style
WS_BORDER
WS_CAPTION
WS_CHILD
WS_GROUP
WS_POPUP
WS_SYSMENU

Force Dialog Templates Option

The DIALOG template allows you to specify CAPTION , CLASS , FONT and STYLE for the dialog itself. If this menu option is checked, STORC will get this information from the form window and put it into the dialog template it writes. However, you will have to register the dialog class to use the resulting template; the custom class dialog style can be improperly interpreted, and resulting dialog will be not usable unless you modify it.

We decided to provide 'default' dialog template that definitely will compile and run from any user's program. Check this option to force the 'standard' template.

Dialog Statement

Screen Related Coordinates Option

There are 2 ways a dialog box can be positioned:

Relative to parent window or

Relative to screen.

The x,y coordinates you specify in the DIALOG statement are interpreted one of those two ways depending on DS_ABSALIGN style. If DS_ABSALIGN is specified, the coordinates are screen-related, and parent-related otherwise.

STORC provides you an option to capture the coordinates either way: if 'Options/Screen-Related Coordinates' is checked, regardless of actual window style STORC forces DS_ABSALIGN flag and calculates dialog box coordinates as screen-based. If this menu option is unchecked, and the window has a parent, STORC clears the DS_ABSALIGN flag and calculates parent-related coordinates. If you want your form to always appear at the same location on the screen, check this option. Otherwise the form will appear at given offset of its parent window.

DS_ABSALIGN

Dialog Statement

Native VBX Option

Microsoft Visual C++ 1.0 allows you to use VBX controls in your C++ program. Microsoft has 'stretched' the dialog CONTROL statement format to accommodate VBX controls: the text field now bears important information on VBX file and control name. Class name is always VBControl.

STORC can either map a VBX control into 'normal' control class you specify, or create a control statement in this 'new' format (if NativeVBX option is checked).

VBX Control Statement Format

Storc Output

DIALOG Scripts Created by STORC
Additional Info

Dialog Scripts Created by Storc

STORC creates the dialog scripts by enumerating the child windows of your form window and enumerating all the child windows of those child windows.

Typically, you will only be interested in those controls which are direct children of your form, but STORC gives you more just for the sake of genericity, printing the whole window hierarchy. This can give interesting side effects for controls like Drop-Down Listbox, that has two child windows: an edit field and a list box. There is no need to include those child windows into your actual script; it is sufficient to define a Drop-Down Listbox since it creates its children automatically.

Then STORC generates the resulting script and inserts into the editor at cursor (possibly replacing current selection). The script generated depends on current options and storc.ini settings for class translation.

Every level in this hierarchy is indented one space in the script, making it easier to understand and modify.

Additional Info

Storc also prints out some additional information that is not normally part of a dialog template. This information is supplied AS A COMMENT, and is introduced by a comment start mark (of information that STORC currently supplies: font information and color information.

Menu Info
Font Info
Color Info

Font Info

Storc provides font information for the whole dialog box, as well as individual controls. Since font size and facename are ALWAYS provided by STORC as a part of dialog templates, additional info such as weight and whether the font is italic.
For individual controls, full font information is printed if appropriate.
Typically a control assumes the exact font of its dialog box. If STORC determines that a particular control's font is somehow different from its parent box font, it supplies full font information for the control as a comment.

If the control has DEFAULT font instead of the font defined its parent dialog box, STORC will indicate so.

If STORC provides no font information for a control, then the control font is exactly the same as its dialog box' font. There needs to be no special processing in this case since the dialog box will create all its controls using the font specified in its template.

Handling Control Fonts in C Code

C Code Example

By default all controls in a dialog box would assume the font of this dialog box. If STORC reports that some control has different font, you will have to handle this case in your C or TurboPascal code. An obvious approach would be to send a WM_SETFONT message to that particular control when the dialog box is created, e.g. on WM_INITDIALOG. If the font of the control is not one of the default system fonts provided by windows, you will have to create a logical font, send its handle to the control in a WM_SETFONT message and then destroy the font when the box is dismissed (on WM_DESTROY). Another approach would be to subclass that particular control and handle the creation, change and destruction of the logical font in the subclass proc.
Here we illustrate the 1st approach:

```
int FAR PASCAL _export SampleDlgBoxProc(

        HWND hWnd,
        UINT iMessage,
        WPARAM wParam,
        LPARAM lParam){
static LOGFONT lFont=0;
static HFONT hfontDlg=0;  //font of the dialog box.
static HFONT hfontControl=0;      //font of the control. Differs from dlg box font (not bold)

switch(iMessage){
        case WM_INITDIALOG:
                 /* Get dialog box font and create version that is not bold. */
                hfontDlg = (HFONT) SendMessage(hdlg, WM_GETFONT, 0, 0L);
                if (hfontDlg){
                        if (GetObject(hfontDlg, sizeof(LOGFONT), (LPSTR)&lFont)) {
                                Font.lfWeight = FW_NORMAL;
                                hfontControl= CreateFontIndirect((LPLOGFONT) &lFont);
                                if (hFontControl)
                                 SendDlgItemMessage(hdlg, ID_CTRL1, WM_SETFONT,
(WPARAM) hfontControl, 0);
                        }
                }
                return TRUE;
        case WM_DESTROY:
                DeleteObject(hfontControl);
                PostQuitMessage(0);
                return 0L;
}
```

Color Info

Storc Gold reads text foreground and background, as well as the background brush information from each control and the dialog box itself. This feature can be optionally set by toggling the **Report Colors** option

The color information is printed as a comment below each control statement and includes RGB values for control foreground/background, brush type and color if applicable

Handling Control Colors in C Code

C Code Example

Standard approach would be to handle WM_CTLCOLOR message in your dialog procedure

```
int FAR PASCAL _export SampleDlgBoxProc(
static HBRUSH hbrGray=0;
switch(msg) {
        case WM_INITDIALOG:

                /* Create a gray brush */

                hbrGray = CreateSolidBrush(RGB(0, 255, 0));
                return TRUE;

        case WM_DESTROY:

                DeleteObject(hbrGray);
                return 0;

        case WM_CTLCOLOR:

         switch(HIWORD(lParam)) {
                case CTLCOLOR_DLG://dialog box itself
                     return (LRESULT) hbrGray;
                case CTLCOLOR_EDIT:
                 /* Set text to red and background to white */
                     SetTextColor     ((HDC) wParam, RGB(255, 0, 0));
                     SetBkColor       ((HDC) wParam, RGB(0, 0, 0));
                     return (LRESULT) hbrGray;
                case CTLCOLOR_MSGBOX:
                 /* For single-line edit controls, this code
                     must be processed so that the background
                     color of the format rectangle will also
                     be painted with the new color.*/

                 return (LRESULT) hbrGray;

                case CTRLCOLOR_STATIC:
                 /* Set text to black and background to gray */
                     SetTextColor((HDC) wParam, RGB(255, 255, 255));
                     SetBkColor   ((HDC) wParam, RGB(192, 192, 192));
                     return (LRESULT) hbrGray;
        }
        return (LRESULT) NULL;
}
}
```

Menu Info

A script created by STORC GOLD includes 2 menu resources:

Main Window Menu
It is always named MENU_1, and a reference to MENU_1 is included in the DIALOG statement

System Menu
It is named SYSMENU_1

Dialog Statement

Disclaimer

PractiSys disclaims all the warranties relating to this software, whether express or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, and all such warranties are expressly and specifically disclaimed.

Neither PractiSys, nor anyone else who has been involved in the creation, production or delivery of this software shall be liable for any indirect, consequential, or incidental damages arising out of use or inability to use such software even if PractiSys has been advised of the possibility of such damages or claims.

In no event shall the PractiSys' liability for any damages ever exceed the price paid for the license to use the software.

Regardless of the form of claim, the person using the software bears all risk as to the quality and performance of the software.

Some states do not allow the exclusion of the limit of liability for consequential or incidental damages, so the above limitation may not imply to you.

The agreement shall be governed by laws of state of California.

Any action or proceeding brought by either party against the other arising out of or related to this agreement shall be brought only in a STATE or FEDERAL COURT of competent jurisdiction located in Ventura County, California.