

```

function ChooseColor(var CC: TChooseColor): Bool;
type
PFindReplace = ^TFindReplace;
TFindReplace = record
  lStructSize: Longint;           { size of this struct $20 }
  hWndOwner: HWND;              { handle to owner's window }
  hInstance: THandle;           { instance handle of .EXE that
                                contains cust. dlg. template }
  Flags: Longint;               { one or more of the fr_?? }
  lpstrFindWhat: PChar;         { ptr. to search string }
  lpstrReplaceWith: PChar;      { ptr. to replace string }
  wFindWhatLen: Word;           { size of find buffer }
  wReplaceWithLen: Word;        { size of replace buffer }
  lCustData: Longint;           { data passed to hook fn. }
  lpfnHook: function (Wnd: HWND; Msg, wParam: Word; lParam: Longint): Word;
                                { ptr. to hook fn. or nil }
                                { custom template name }
  lpTemplateName: PChar;
end;

function ReplaceText(var FindReplace: TFindReplace): HWND;
type
PChooseFont = ^TChooseFont;
TChooseFont = record
  lStructSize: Longint;           { }
  hWndOwner: HWND;              { caller's window handle }
  hDC: HDC;                     { printer DC/IC or nil }
  lpLogFont: PLogFont;          { ptr. to a LOGFONT struct }
  iPointSize: Integer;          { 10 * size in points of selected font }
  Flags: Longint;               { enum. type flags }
  rgbColors: Longint;           { returned text color }
  lCustData: Longint;           { data passed to hook fn. }
  lpfnHook: function (Wnd: HWND; Msg, wParam: Word; lParam: Longint): Word;
                                { ptr. to hook function }
                                { custom template name }
                                { instance handle of .EXE that contains cust.
                                dlg. template }
  lpTemplateName: PChar;        { return the style field here must be lf_FaceSize
                                or bigger }
  hInstance: THandle;           { same value reported to the EnumFonts call back
                                with the extra fonttype_bits added }
  lpszStyle: PChar;             { minimum pt size allowed & }
  nFontType: Word;              { max pt size allowed if cf_LimitSize is used }
  nSizeMin: Integer;
  nSizeMax: Integer;
end;

{ these are extra nFontType bits that are added to what is returned to the EnumFonts callback routine }

Italic_FontType = $0200;
Regular_FontType = $0400;

wm_ChooseFont_GetLogFont = wm_User + 1;

{ strings used to obtain unique window message for communication between dialog and caller }

SetRGBString = 'commdlg_SetRGBColor';
FindMsgString = 'commdlg_FindReplace';
HelpMsgString = 'commdlg_help';

{ HIWORD values for lParam of commdlg_LBSELChangeNotify message }

cferr_NoFonts = $2001;
cferr_MaxLessThanMin = $2002;

fnErr_FilenameCodes = $3000;
fnErr_SubclassFailure = $3001;
fnErr_InvalidFilename = $3002;
fnErr_BufferTooSmall = $3003;

frErr_FindReplaceCodes = $4000;
frErr_BufferLengthZero = $4001;

ccErr_ChooseColorCodes = $5000;

{

Digital and the DIGITAL logo are registered trademarks of
Digital Equipment Corporation.

}

```